

Diseño conceptual de bases de datos

UN ENFOQUE DE ENTIDADES-INTERRELACIONES

BATINI
CERI
NAVATHE



ADDISON-WESLEY/DIAZ DE SANTOS

Diseño conceptual de bases de datos

Un enfoque de entidades-interrelaciones

Carlo Batini

Universitá di Roma «La Sapienza»

Stefano Ceri

Politecnico di Milano

Shamkant B. Navathe

Georgia Institute of Technology

Versión en español de

Víctor Martín García

Diego Romero Ibancos

Universidad Pontificia de Salamanca

Campus de Madrid. España

Con la colaboración de:

Luis Joyanes Aguilar

Universidad Pontificia de Salamanca

Campus de Madrid, España

Américo Vargas Villazón

Heuresis, SRL

La Paz, Bolivia



ADDISON-WESLEY/DIAZ DE SANTOS

Argentina · Brasil · Chile · Colombia · Ecuador · España

Estados Unidos · México · Perú · Puerto Rico · Venezuela

Versión en español de la obra titulada ,*Conceptual Database Design: An Entity-Relationship Approach*, de *Carlo Batini, Stefano Ceri, Shamkant B.Navarthe*, publicada originalmente en inglés por Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, E.U.A. © 1992

Esta edición en español es la única autorizada.

Copublicación de Addison-Wesley Iberoamericana, S.A.
y Ediciones Díaz de Santos, S.A.

© 1994 por **Addison-Wesley Iberoamericana, S.A.**
Wilmington, Delaware, E.U.A.

Reservados todos los derechos.

«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»

Impreso en Estados Unidos de América. Printed in U.S.A.

ISBN: 0-201-60120-6

1 2 3 4 5 6 7 8 9 10-MA-99 98 97 96 95 94

A mis padres: Curzio, fallecido en 1975, y Laura, fallecida quince años después.

CARLO BATINI

A mis padres: Luciana Arcidiacono y Mauro Ceri.

STEFANO CERI

A mis padres: Bhalchandra y Vijaya Navathe, por su aliento y apoyo.

SHAMKANT B. NAVATHE

Contenido

Primera parte. Diseño conceptual de bases de datos	1
1. Una introducción al diseño de bases de datos	3
2. Conceptos sobre el modelado de datos	17
3. Metodología para el diseño conceptual	65
4. Diseño de vistas	99
5. Integración de vistas	137
6. Cómo mejorar la calidad de un esquema de base de datos	159
7. Documentación y mantenimiento de esquemas	193
Segunda parte. Análisis funcional en el diseño de bases de datos	221
8. Análisis funcional usando el modelo de flujo de datos	223
9. Análisis conjunto de datos y funcional	257
10. Estudio de un caso	277
Tercera parte. Diseño lógico y herramientas de diseño	307
11. Diseño lógico de alto nivel usando el modelo de entidades-interrelaciones	309
12. Diseño lógico en el modelo relacional	351
13. Diseño lógico en el modelo de redes	395
14. Diseño lógico en el modelo jerárquico	429
15. Herramientas de diseño de bases de datos	465
Indice onomástico	533
Indice analítico	537

Índice general

Primera parte. Diseño conceptual de bases de datos	1
1. Una introducción al diseño de bases de datos	3
El diseño de bases de datos en el ciclo de vida de los sistemas de información	4
Eases del diseño de bases de datos	7
Interacción entre el diseño de bases de datos y el análisis funcional	9
Modelos y herramientas para el diseño de bases de datos y el análisis funcional	11
¿Por qué es valioso el diseño conceptual?	12
Resumen	13
2. Conceptos sobre el modelado de datos	17
Abstracciones en el diseño conceptual de bases de datos	17
Propiedades de las correspondencias entre las clases	22
Modelos de datos	30
El modelo de entidades-interrelaciones	35
Lectura de un diagrama de entidades-interrelaciones	54
Resumen	55
3. Metodología para el diseño conceptual	65
Primitivas del diseño conceptual	66
Estrategias para el diseño de esquemas	77
Criterios de elección entre conceptos	89
Entradas, salidas y actividades del diseño conceptual	90
Resumen	93
4. Diseño de vistas	99
Diseño de vistas a partir de requerimientos expresados en lenguaje natural	100
Diseño de vistas a partir de formularios	107

Diseño de vistas a partir de formatos de registros	117
Resumen	126
5. Integración de vistas	137
Aspectos de la integración de vistas	138
Integración de vistas a gran escala	140
Análisis y resolución de conflictos	142
Fusión de vistas	148
Resumen	151
6. Cómo mejorar la calidad de un esquema de base de datos	159
Cualidades de un esquema de base de datos	160
Transformaciones de esquemas	165
Transformaciones para lograr minimalidad	166
Transformaciones para lograr expresividad y autoexplicación ..	169
Transformaciones para lograr normalización	172
Ejemplo de reestructuración de un esquema	185
Resumen	188
7. Documentación y mantenimiento de esquemas	193
Una metodología para documentar esquemas conceptuales	194
Mantenimiento de esquemas conceptuales	203
Estructura y diseño de un diccionario de datos	209
Resumen	216
Segunda parte. Análisis funcional en el diseño de bases de datos	221
8. Análisis funcional usando el modelo de flujo de datos	223
El modelo de flujo de datos para el análisis funcional	224
Primitivas para el análisis funcional	227
Estrategias para el análisis funcional	229
Diseño de un esquema funcional	237
Cualidades de un esquema funcional	239
Documentación y mantenimiento de un esquema funcional ...	242
Resumen	243
Apéndice. Repaso de modelos para el análisis funcional	247
9. Análisis conjunto de datos y funciones	257
Esquemas externos para los diagramas de flujo de datos	258
Una metodología para el análisis conjunto de datos y funcional	260
Sugerencias para refinamientos mutuos	267
Esquemas de navegación para operaciones con bases de datos ..	270
Resumen	274

<u>10. Estudio de un caso</u>	277
<u>Requerimientos</u>	277
<u>Esquemas de armazón</u>	279
<u>Primer refinamiento</u>	282
<u>Segundo refinamiento</u>	291
<u>Verificación de compleción</u>	296
<u>Esquemas de navegación</u>	298
<u>Resumen</u>	303
 Tercera parte. Diseño lógico y herramientas de diseño	 307
<u>11. Diseño lógico de alto nivel usando el modelo de entidades-interrelaciones</u>	309
<u>Un enfoque global del diseño lógico</u>	310
<u>Modelado de la carga de bases de datos</u>	313
<u>Decisiones sobre datos derivados</u>	317
<u>Eliminación de jerarquías de generalización</u>	320
<u>Partición de entidades</u>	327
<u>Fusión de entidades e interrelaciones</u>	331
<u>Selección de claves primarias</u>	332
<u>Diseño lógico de alto nivel en el caso de estudio</u>	334
<u>Resumen</u>	344
 <u>12. Diseño lógico en el modelo relacional</u>	 351
<u>El modelo relacional</u>	352
<u>Correspondencia de esquemas del modelo ER al modelo relacional</u>	356
Correspondencia de operaciones del modelo ER al modelo relacional	369
Base de datos del caso de estudio en el modelo relacional	369
<u>Retroingeniería de los esquemas relacionales a esquemas ER ...</u>	374
<u>Resumen</u>	388
 <u>13. Diseño lógico en el modelo de redes</u>	 395
<u>El modelo de redes</u>	396
<u>Correspondencia de esquemas del modelo ER al modelo de redes</u>	401
Correspondencia de operaciones del modelo ER al modelo de redes	411
Base de datos del caso de estudio en el modelo de redes	414
<u>Retroingeniería de los esquemas de redes a esquemas ER</u>	418
<u>Resumen</u>	422

14. Diseño lógico en el modelo jerárquico	429
El modelo jerárquico	430
Correspondencia de esquemas del modelo ER al modelo jerárquico	437
Correspondencia de operaciones del modelo ER al modelo jerárquico	447
Base de datos del caso de estudio en el modelo jerárquico	450
Retroingeniería de esquemas jerárquicos a esquemas ER	455
Resumen	459
15. Herramientas de diseño de bases de datos	465
Ingeniería de software asistida por computador	466
Características convenientes en un sistema de diseño de bases de datos	467
Falta de concordancia entre metodologías y herramientas	473
Herramientas básicas para el diseño conceptual	475
Herramientas básicas para el diseño lógico	481
Herramientas actuales de diseño de bases de datos orientadas a la investigación	482
Herramientas actuales de diseño de bases de datos comerciales	488
Herramientas CASE comerciales actuales que permiten diseñar bases de datos	494
Resumen de herramientas comerciales	499
Limitaciones de los entornos de diseño automatizado de bases de datos	505
Tendencias en los entornos de diseño automatizado de bases de datos	506
Vocabulario técnico bilingüe	517
Indice onomástico	533
Indice analítico	537

Prefacio

Antecedentes

El diseño de bases de datos es el proceso por el que se determina la organización de una base de datos, incluidos su estructura, contenido y las aplicaciones por desarrollar. Durante mucho tiempo, el diseño de bases de datos fue considerado una tarea para expertos: más un arte que una ciencia. Sin embargo, se ha progresado mucho en el diseño de bases de datos y éste se considera ahora una disciplina estable, con métodos y técnicas propios. Debido a la creciente aceptación de las bases de datos por parte de la industria y el gobierno en el plano comercial, y a una variedad de aplicaciones científicas y técnicas, el diseño de bases de datos desempeña un papel central en el empleo de los recursos de información de la mayoría de las organizaciones. El diseño de bases de datos ha pasado a constituir parte de la formación general de los informáticos, en el mismo nivel que la capacidad de construir algoritmos usando un lenguaje de programación convencional.

El diseño de bases de datos se realiza por lo general en tres fases. La primera fase, llamada de *diseño conceptual*, produce una representación abstracta y de alto nivel de la realidad. La segunda fase, llamada de *diseño lógico*, convierte esta representación en especificaciones que pueden implantarse en un sistema de cómputo y ser procesadas por él. La tercera fase, llamada de *diseño físico*, determina las estructuras de almacenamiento físico y los métodos de consulta requeridos para un acceso eficaz a los contenidos de una base de datos a partir de dispositivos de almacenamiento secundario.

Este libro trata las dos primeras fases del diseño de bases de datos, con una marcada orientación hacia los asuntos relacionados con el usuario y con la aplicación, más que hacia un sistema y entorno específicos de software/hardware. Las fases del diseño lógico y conceptual pueden desarrollarse *independientemente* de la elección de un sistema de administración de bases de datos (DBMS, *database management system*) específico. Por tanto, se supone el conocimiento de los conceptos generales de bases de datos, o bien que la mayoría de los lectores tienen alguna experiencia con los DBMS.

Creemos que una ejecución sistemática y minuciosa de estas primeras fases de diseño rinde un gran beneficio a largo plazo. En realidad, muchas organizaciones se están dando cuenta de la necesidad de crear un diseño lógico y conceptual, al mismo tiempo que se inclinan hacia la tecnología de bases de datos relacional y orientada a objetos.

En este libro se usa el modelo *entidades-interrelaciones (EI)* de Chen, con algunas añadiduras necesarias para una mejor representación conceptual. Este modelo se usa ampliamente en muchas metodologías de diseño como una representación gráfica eficaz, y es el estándar «de facto» para la mayoría de las herramientas automáticas de apoyo al diseño de bases de datos. Si bien este libro se centra en el diseño conceptual de bases de datos, se muestra una metodología para la realización conjunta del diseño conceptual de bases de datos y el análisis funcional. Este enfoque mixto se basa en técnicas bien conocidas, comunes a ambos enfoques.

Propósito del libro

Los objetivos principales de este libro son los siguientes:

- Ofrecer un tratamiento minucioso y sistemático del diseño lógico y conceptual.
- Basar este tratamiento en el muy acogido modelo de entidades-interrelaciones.
- Recomendar que el diseño conceptual y el análisis funcional se desarrollen a la par.
- Tratar de manera completa la conversión del diseño conceptual en el modelo de entidades-interrelaciones a los tres modelos populares de bases de datos: relacional, de redes y jerárquico. Asimismo, se presenta la problemática de la retroingeniería desde estos tres modelos al modelo ER.
- Ilustrar los conceptos mediante un caso de estudio realista y extenso.
- Ofrecer una visión general de la última tecnología en el campo de las herramientas de diseño.
- Ofrecer suficiente apoyo pedagógico para los estudiantes de este tema, en términos de ejercicios y notas bibliográficas sobre la literatura pertinente.

A quién va dirigido

La actividad más importante en el diseño conceptual es entender y modelar la realidad; esta tarea es difícil y comúnmente sólo es desarrollada por expertos. Una vez asimilada, la labor del diseño lógico es bastante simple y directa. El objetivo principal de este libro es tratar el diseño conceptual, no sólo para be-

neficio de los expertos, sino para introducirlo a un público más amplio compuesto de:

1. Estudiantes, que reclaman un tratamiento preciso y riguroso del diseño conceptual y lógico de bases de datos para completar un primer curso sobre modelos y sistemas de bases de datos.
2. Profesionales en ejercicio (administradores de bases de datos, analistas, consultores y programadores de bases de datos), quienes harán uso de este material para formalizar y resolver problemas de diseño de bases de datos, con frecuencia mal definidos. Ciertamente, la metodología presentada en este libro puede aplicarse a la mayoría de los diseños y, así, ayudar a los diseñadores a resolver sus problemas de diseño de forma sistemática.
3. Usuarios de bases de datos, quienes necesitan una cultura básica sobre el tema para comunicarse con el personal de administración de bases de datos y poder especificar sus necesidades. Asimismo, serán capaces de observar y controlar el proceso de diseño y entender el significado y la estructura de la base de datos almacenada en su sistema de información.

Los contenidos de este libro son autónomos: todos los conceptos se definen antes de ser usados. Sin embargo, el libro no incluye una descripción de las características de los sistemas de bases de datos o de los lenguajes que pueden usarse en la programación de sistemas de bases de datos. Por esto, es requisito previo algún conocimiento sobre sistemas de bases de datos, obtenido de un primer curso o de la experiencia en el empleo concreto de las bases de datos. Se recomienda el uso del libro *Fundamentals of Database Systems*, de Elmasri y Navathe (Benjamin/Cummings, 1989), como una fuente completa de material de referencia. Los capítulos 12, 13 y 14 proporcionan introducciones que resumen los modelos de datos relacional, de redes y jerárquico, respectivamente.

Perfil del libro

El libro se divide en tres partes, precedidas por un capítulo introductorio. La última parte concluye con un capítulo de David Reiner sobre herramientas de diseño.

La primera parte sigue un enfoque orientado a los datos y trata el diseño conceptual de bases de datos como independiente del diseño de aplicaciones.

El primer capítulo ilustra el papel del diseño de bases de datos dentro del ciclo de vida de los sistemas de información, y la diferencia entre el enfoque orientado a los datos y el orientado a las funciones en el diseño de sistemas de información. El capítulo 2 presenta los conceptos de modelado de datos y, específicamente, el modelo ER, para que, después de completarlo, el lector sea capaz de entender los esquemas ER. El capítulo 3 muestra primitivas de diseño y

estrategias para diseñar esquemas ER. Al final de este capítulo, el lector debe ser capaz de *construir* pequeños esquemas ER.

El capítulo 4 se divide en tres apartados; cada uno ilustra enfoques específicos del diseño conceptual basados en las diferentes formas de expresión inicial de los requerimientos: descripciones textuales, formularios y formatos de registro en Cobol. Cada apartado puede leerse independientemente. El capítulo 5 describe cómo diferentes esquemas deben *integrarse* para generar un único esquema global. El capítulo 6 muestra cómo un esquema conceptual debe *reestructurarse* para perfeccionar sus propiedades (incluidas compleción o integridad, minimalidad, expresividad, legibilidad y normalización). El capítulo 7 muestra cómo se debe *documentar* el diseño conceptual, reuniendo diversas descripciones de diseño, y cómo esa documentación puede utilizarse en el mantenimiento de una base de datos y en la integración del diccionario de datos.

La segunda parte sigue un enfoque conjunto orientado a los datos y a las funciones, e integra el modelado conceptual con el análisis funcional. El capítulo 8 trata el *análisis funcional*, al introducir el modelo de flujo de datos y al mostrar primitivas y estrategias de diseño. El capítulo 9 ilustra el *enfoque conjunto orientado a datos y funciones* para el diseño conceptual de datos y funciones; este método produce una especificación de alto nivel de las *operaciones de bases de datos* que servirá como guía para el consiguiente diseño lógico y físico de las bases de datos.

El capítulo 10 contiene un extenso estudio de caso. Se presenta un ejemplo bastante realista de una compañía de autobuses que ofrece una variedad de excursiones. Se ha procurado exponer los aspectos de la compañía de autobuses que son pertinentes en el diseño de la base de datos.

La tercera parte del libro se refiere al diseño lógico, que es el proceso de convertir el diseño conceptual en una estructura de base de datos realizable dentro de un sistema específico de gestión de bases de datos. Primero se muestra el diseño independiente del modelo, es decir, se consideran transformaciones que simplifican el esquema conceptual sin tener en cuenta el modelo de datos que constituye el objetivo final. Después se considera la correspondencia entre el esquema conceptual y cada uno de los tres modelos de datos más destacados: relacional, de redes y jerárquico.

El capítulo 11 estudia el *diseño lógico independiente del modelo*, usando el modelo entidades-interrelaciones, y describe las transformaciones iniciales del esquema conceptual en un esquema simplificado, intermedio, entre conceptual y lógico. Los próximos tres capítulos usan este esquema simplificado como punto de partida de transformaciones subsecuentes para adaptarlo a las familias predominantes de DBMS, implantadas comercialmente.

Los capítulos 12 al 14 convierten un esquema conceptual del modelo ER en un esquema *relacional*, un esquema de *redes* (DBTG o CODASYL) y un esquema *jerárquico*, respectivamente. En estos tres capítulos se han resumido las características, limitaciones y lenguajes esenciales asociados con estos modelos. En cada uno de esos capítulos se examinan dos problemas adicionales. Primero, se considera la traducción de las operaciones del esquema conceptual a lenguajes

apropiados de manipulación de datos para el modelo de datos objetivo (relacional, de redes o jerárquico). Estos se aclaran mediante ejemplos de operaciones sobre la base de datos del caso de estudio. En segundo lugar, se aborda el problema de la *retroingeniería* (o *ingeniería inversa*) para cada modelo, de modo que los esquemas de bases de datos existentes en los respectivos modelos puedan abstraerse o ser analizados para obtener un esquema conceptual.

El capítulo 15, sobre las herramientas de diseño de bases de datos, lo aporta David Reiner. Primero se analizan los asuntos generales, incluyendo la arquitectura y las características deseadas de las herramientas de diseño y, luego se describen algunas de las herramientas para el diseño de bases de datos asistido por computador actualmente disponibles como prototipos de investigación o en el mercado.

La estructura del libro, representada en el cuadro anexo, indica las relaciones de precedencia entre los capítulos y sugiere varias secuencias de lectura.

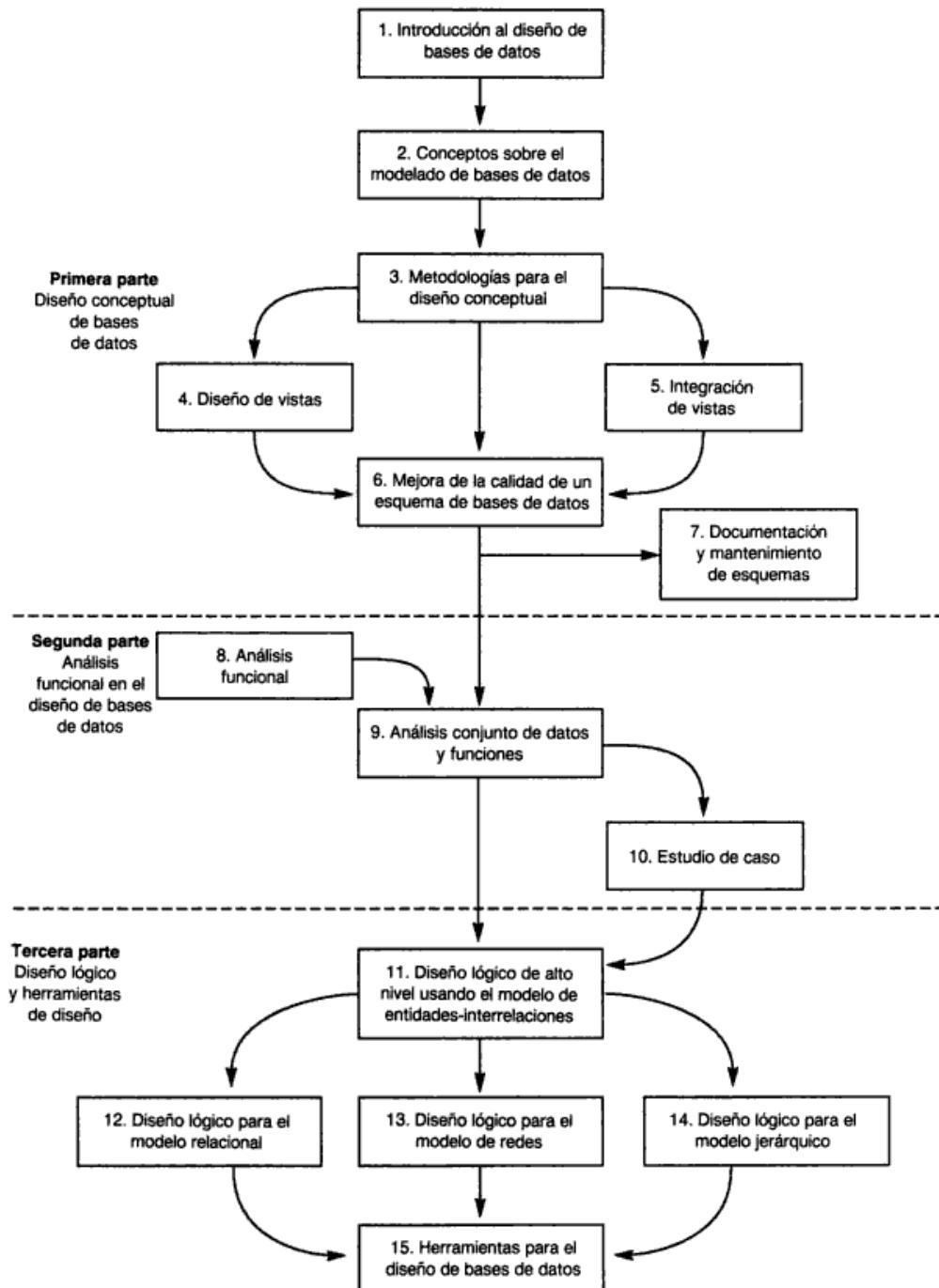
Reconocimientos

Nuestro enfoque hacia el diseño de bases de datos acusa una marcada influencia del Proyecto Dataid, desarrollado en Italia entre los años 1980 y 1985, y patrocinado por el Consejo Nacional de Investigación. Damos las gracias a los colegas que participaron activamente en el proyecto, particularmente a Antonio Albano, Valeria de Antonellis y Antonio di Leva.

Cada uno de los tres autores ha investigado, por separado, sobre el diseño de bases de datos, junto con otros colegas. En particular, Carlo Batini desea dar las gracias a Maurizio Lenzerini, Giuseppe Di Battista y Giuseppe Santucci.

Stefano Ceri desea reconocer la cooperación de Giampio Bracchi, Giuseppe Pelagatti, Barbara Pernici, Paola Mostacci y Federico Barbic durante los años de actividad del Proyecto Dataid, patrocinado por el Consejo Nacional de Investigación y por la Fundación Nacional de Ciencias. Varios estudiantes del Politecnico di Milano han investigado sobre el diseño de bases de datos con él, como parte de sus tesis; desea mencionar de forma especial a Alessandro Rivelilla, Paolo Pellegrini, Corrado Luppi, Marco Spizzi y Giovanni Zorzino. Asimismo, da las gracias por su cooperación a Gio Wiederhold por el acercamiento, junto con Sham Navathe y Barbara Pernici, hacia algunos temas de investigación sobre el diseño de bases de datos, como por ejemplo el diseño de fragmentación y localización de datos; y reconoce también la cooperación de Barbara Pernici, Giampio Bracchi, Letizia Tanca, Fabio Schreiber, Giuseppe Pelagatti y Licia Sabattella en el ofrecimiento de cursos avanzados de diseño de bases de datos.

Sham Navathe desea reconocer la colaboración de Jim Larson, Ramez Elmasri, Amit Sheth, Skip Pumfrey y Alan Simon en proyectos conjuntos de diseño de bases de datos y herramientas de diseño, que realizó con las corporaciones Honeywell, Bellcore y Digital. Desea agradecer a la Fundación Nacional de



Relaciones de precedencia entre los capítulos de este libro.

Ciencias el otorgamiento de las subvenciones de investigación EE.UU.- Italia que facilitaron la colaboración con científicos italianos, incluido el desarrollo de este libro. Aparte de a los coautores de esta obra, desea expresar su agradecimiento a Maurizio Lenzerini, Tiziana Catarci, Giuseppe Di Battista, Giampio Bracchi, Barbara Demo y Barbara Pernici, así como a otros miembros del Proyecto Dataid. Entre los estudiantes graduados de la University of Florida, desea mencionar la influencia del trabajo de Tarek Anwar, Aloysius Cornelio, Sunit Gala y Ashoka Savasere sobre sus propios trabajos en el campo del diseño de bases de datos.

Versiones preliminares de este libro han sido analizadas a lo largo de varias ediciones de talleres, cursos y conferencias didácticas; las metodologías se han usado en un contexto profesional. En particular, Stefano Ceri dio un curso en el Centro de Investigación de IBM en Río de Janeiro en 1987; Carlo Batini presentó una introducción al tema en la Conferencia Internacional sobre Tecnología Ampliada de Bases de Datos (EDBT) en Venecia en 1988; Stefano Ceri y Sham Navathe presentaron una introducción en la Conferencia Internacional sobre Grandes Bases de Datos (VLDB) en Los Angeles en 1989.

Varios colegas han contribuido en la preparación del manuscrito. Agradecemos en particular a Jim Larson, quien ha hecho comentarios eficaces sobre las dos primeras partes del libro; sus observaciones detalladas han influido mucho en la organización de esta revisión. Mary Loomis aportó indicaciones útiles para la expansión y mejora de la primera versión. Agradecemos también a Maurizio Lenzerini, Mike Mannino y Ashok Malhotra sus útiles comentarios. Young-chul Oh ayudó en la conversión de manuscritos. Kalamakar Karlappalem y Magdi Morsi colaboraron en la corrección de pruebas. Se agradece mucho la temprana cooperación de Alan Apt y Mary Ann Telatnik en el desarrollo de este libro. Nick Murray hizo un trabajo espléndido como editor. Ciertamente, todos los errores y omisiones del libro son sólo responsabilidad de los autores, quienes se presentan en orden alfabético.

Para terminar, Sham Navathe desea reconocer el apoyo y sacrificio de su esposa Aruna y de sus hijos Manisha y Amol en todo el proceso de creación de este libro. Se agradece enormemente la ayuda de ellos en la preparación del índice. Stefano Ceri desea dar las gracias a Teresa y Paolo por su ánimo y apoyo durante la preparación del libro.

Primera parte

Diseño conceptual de bases de datos*

El objetivo del diseño conceptual es crear un esquema conceptual de alto nivel, independiente del DBMS, partiendo de especificaciones de requerimientos que describan la realidad. Los conceptos de diseño de bases de datos se presentan por medio de la descripción de los *mecanismos de abstracción*: procesos mentales mediante los cuales nos concentraremos en las propiedades comunes de los datos, sin atender a los detalles no pertinentes. Luego se presenta el *modelo de entidades-interrelaciones*, usado a lo largo del libro, y se muestra cómo este modelo utiliza los mecanismos de abstracción.

A continuación se describen las *metodologías de diseño*. En el proceso de diseño, se propone tomar todas las decisiones de diseño de forma estructurada mediante la aplicación sistemática de las primitivas de refinamiento. Cada primitiva se aplica a una descripción inicial de la realidad y la transforma en una nueva y más rica descripción; las primitivas se clasifican en descendentes y ascendentes. Las *estrategias* para el refinamiento de un esquema emplean indistintamente las primitivas descendentes, ascendentes o un enfoque mixto. Así, una descripción inicial del esquema, llamada *esquema de armazón*, evoluciona hasta el esquema conceptual final.

El diseño de un esquema está, asimismo, influido por los tipos de requerimientos iniciales disponibles. Como recursos alternativos, se consideran las *descripciones textuales en lenguaje natural*, los for-

* El término «relationship» se ha traducido por «interrelación», aunque en España está también muy extendido en la jerga informática el término relación.

A lo largo del texto se conservan las siglas ER originales en inglés por su amplio uso en la comunidad informática internacional y española (abreviatura de Entidad/Relación), pese a que la coherencia gramatical obligaría a utilizar EI.

mularios y los formatos de registros en Cobol; para cada uno de ellos se sugieren diferentes enfoques de diseño.

Las consideraciones anteriores suponen que el dominio global de la aplicación puede modelarse mediante un único esquema. Para las aplicaciones de grandes bases de datos, es conveniente dividir el dominio de la aplicación en varios subesquemas o *vistas*. Una vista es un subconjunto del dominio de la aplicación asociado con el punto de vista de un usuario específico. Si se diseña cada vista independientemente, entonces los diferentes subesquemas conceptuales resultantes de cada diseño deben integrarse. Los problemas de la *integración de vistas* se deben a conflictos entre esas vistas que reflejan perspectivas distintas de los mismos datos en diferentes contextos.

El esquema conceptual final puede ser sometido a *reestructuración*. Las cualidades de un esquema conceptual están formalmente definidas; incluyen compleción, corrección, minimalidad, expresividad, legibilidad, autoexplicación y extensibilidad. Además, incluyen *normalidad*, una propiedad que ha sido definida formalmente en el contexto del modelo de datos relacional y que se generaliza para los esquemas de entidades-interrelaciones. Cada característica se analiza por separado y se sugieren transformaciones para mejorarla.

Después de completar el diseño conceptual, se sugiere la creación de una *documentación completa*, que incluya varios esquemas y especificaciones; estos documentos pueden usarse durante la operación con la base de datos para el mantenimiento de esquemas y programas, y también se incluyen en el diccionario integrado de datos.

Una introducción al diseño de bases de datos

Las últimas dos décadas se han caracterizado por un fuerte crecimiento en el número e importancia de las aplicaciones de bases de datos. Las bases de datos son componentes esenciales de los sistemas de información, usadas rutinariamente en todos los computadores, desde los supercomputadores intercomunicados hasta los computadores medianos o pequeños. El diseño de bases de datos se ha convertido en una actividad popular, desarrollada no sólo por profesionales sino también por no especialistas.

A finales de la década de 1960, cuando las bases de datos entraron por primera vez en el mercado de software, los diseñadores de bases de datos actuaban como artesanos, con herramientas muy primitivas: diagramas de bloques y estructuras de registros eran los formatos comunes para las especificaciones, y el diseño de bases de datos se confundía frecuentemente con la implantación de las bases de datos. Esta situación ahora ha cambiado: los métodos y modelos de diseño de bases de datos han evolucionado paralelamente con el progreso de la tecnología en los sistemas de bases de datos. Se ha entrado en la era de los sistemas relacionales de bases de datos, que ofrecen poderosos lenguajes de consulta, herramientas para el desarrollo de aplicaciones e interfaces amables con los usuarios. La tecnología de bases de datos cuenta ya con un marco teórico, que incluye la teoría relacional de datos, procesamiento y optimización de consultas, control de concurrencia, gestión de transacciones y recuperación, etc.

Según ha avanzado la tecnología de bases de datos, así se han desarrollado las metodologías y técnicas de diseño. Se ha alcanzado un consenso, por ejemplo, sobre la descomposición del proceso de diseño en fases, sobre los principales objetivos de cada fase y sobre las técnicas para conseguir estos objetivos. Este libro se origina a partir de nuestra creencia de que la mayoría de los conceptos relevantes en el diseño de bases de datos se han establecido con firmeza, y es tiempo de que estos conceptos tengan mayor divulgación.

Desafortunadamente, las metodologías de diseño de bases de datos no son

muy populares; la mayoría de las organizaciones y de los diseñadores individuales confía muy poco en las metodologías para llevar a cabo el diseño y esto se considera, con frecuencia, una de las principales causas de fracaso en el desarrollo de los sistemas de información. Debido a la falta de enfoques estructurados para el diseño de bases de datos, a menudo se subestiman el tiempo o los recursos necesarios para un proyecto de bases de datos, las bases de datos son inadecuadas o inefficientes en relación a las demandas de la aplicación, la documentación es limitada y el mantenimiento es difícil.

Muchos de estos problemas se deben a la falta de una claridad que permita entender la naturaleza exacta de los datos, a un nivel conceptual y abstracto. En muchos casos, los datos se describen desde el comienzo del proyecto en términos de las estructuras finales de almacenamiento; no se da peso a un entendimiento de las propiedades estructurales de los datos que sea independiente de los detalles de la realización. El principal objetivo de nuestro libro es acentuar la importancia de un enfoque conceptual para el diseño de bases de datos. Este mensaje sencillo, aunque importante, está dirigido tanto a los profesionales como a los no especialistas, y es útil en pequeños y grandes proyectos de diseño de bases de datos.

En este capítulo introductorio se expone la importancia de un enfoque conceptual para el diseño de bases de datos y se presenta el diseño de bases de datos como una actividad esencial en el desarrollo de los sistemas de información. Después se ilustra cómo el diseño de bases de datos consta de tres fases discretas, llamadas de diseño *conceptual, lógico y físico*, y se muestra cómo estas fases interactúan con el análisis funcional desde una perspectiva amplia de ingeniería de software. Finalmente, se expone la importancia del diseño conceptual dentro de este marco metodológico.

1.1. El diseño de bases de datos en el ciclo de vida de los sistemas de información

El **sistema de información** de una empresa es un conjunto de actividades que regulan la distribución y el compartimiento de la información, y el almacenamiento de los datos relevantes para la administración de la empresa. (Nos interesan los sistemas de información basados en un computador.) Una **base de datos** es cualquier conjunto grande de datos estructurados almacenado dentro de un computador. Los **sistemas de gestión de bases de datos** (DBMS) son paquetes de software para la gestión de las bases de datos; en particular, para almacenar, manipular y recuperar datos en un computador. Las bases de datos son sólo uno de los componentes de los sistemas de información, que también incluyen programas de aplicación, interfaces para usuarios y otros tipos de paquetes de software. Sin embargo, las bases de datos son esenciales para la supervivencia de cualquier organización, porque los datos estructurados constituyen un recurso



Figura 1.1. Ciclo de vida de los sistemas de información.

esencial para todas las organizaciones, incluidas no sólo las grandes empresas sino también pequeñas compañías y usuarios individuales.

El diseño de bases de datos se sitúa en una perspectiva adecuada al considerarlo dentro del ciclo de vida de los sistemas de información. El **diseño de un sistema de información** es una actividad complicada, que incluye la planificación, especificación y desarrollo de cada componente del sistema. El típico desglose del ciclo de vida de un sistema de información, mostrado en la figura 1.1, incluye un estudio de factibilidad, recolección y análisis de requerimientos, diseño, elaboración de prototipos, implantación, validación y prueba y operación.

Estudio de factibilidad. El estudio de factibilidad trata de determinar la rentabilidad de las distintas alternativas de diseño del sistema de información y las prioridades de los diversos componentes del sistema.

Recolección y análisis de requerimientos. La recolección de requerimientos y su análisis se ocupan de la comprensión de la, así llamada, *misión del sistema de información*, es decir, las áreas de aplicación del sistema dentro de una empresa y los problemas que el sistema debe resolver. Esta fase centra su atención en la interacción con los usuarios del sistema de información. Los usuarios describen sus necesidades a los diseñadores y esas descripciones se reúnen en lo que se llama las *especificaciones de requerimientos*. En general, las **especificaciones de requerimientos** son más bien informales y desorganizadas; se expresan, por lo común, en lenguaje natural o en lenguajes semiformales (por ejemplo, combinación de palabras clave y lenguaje natural).

Diseño. El diseño se ocupa de la especificación de la estructura del sistema de información. Se distingue entre el **diseño de bases de datos** y el **diseño de aplicaciones**. El primero es el diseño de la estructura de la base de datos; el segundo es el diseño de los programas de aplicación. Ambas actividades de diseño son muy complejas y pueden dividirse aún en fases, como se mostrará en los apartados siguientes.

Creación de prototipos. La creación de prototipos es un añadido reciente al ciclo de vida. La mayoría de los paquetes de software incluyen actualmente herramientas para un rápido desarrollo de prototipos, incluyendo los llamados **lenguajes de cuarta generación**. Con estas herramientas, un diseñador puede producir con eficiencia un prototipo del sistema de información o de algunas de sus partes. Un **prototipo** es una realización simplificada, quizás poco eficiente, que se produce para verificar en la práctica que las fases anteriores de diseño se condujeron de forma correcta. El prototipo permite a los usuarios verificar si el sistema de información satisface sus necesidades. Un prototipo que funciona es útil para la corrección o adición de requerimientos sobre la base de la experimentación.

Implantación. La implantación o realización se refiere a la programación de la versión final y operativa del sistema de información. En esta etapa, las alternativas de realización se verifican con cuidado y se comparan, para así lograr que el sistema final cumpla los requerimientos en cuanto a rendimiento.

Validación y prueba. Validación y prueba es el procedimiento mediante el cual se garantiza que cada fase del proceso de desarrollo es de calidad aceptable y es una evolución correcta de la fase anterior. Esto implica verificar si la aplicación refleja las especificaciones de diseño.

Operación. La operación empieza con la carga inicial de datos y termina cuando el sistema se vuelve obsoleto y tiene que ser reemplazado. Durante la operación, se necesita el mantenimiento para hacer que el sistema se adapte a nuevas condiciones, mejorarlo con nuevas funciones o corregir errores no detectados durante la validación.

El ciclo de vida es más que nada un marco de referencia: en muchos proyectos de diseño reales, las diferencias entre las fases son a veces poco claras, algunas fases no se desarrollan y se requiere un gran nivel de retroalimentación entre las fases para mejorar y corregir los resultados de fases anteriores. Sin embargo, el ciclo de vida plantea que al diseño de bases de datos le debe anteceder un análisis de las necesidades, debe conducirse en paralelo con el diseño de aplicaciones y le debe seguir la implantación ya sea de un prototipo o de un sistema final. Ahora es posible concentrar la atención en las fases del diseño de bases de datos.

1.2. Fases del diseño de bases de datos

El diseño de una base de datos es un proceso complejo que abarca varias decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de éstos independientemente, usando métodos y técnicas específicas. El diseño de bases de datos se descompone en diseño conceptual, diseño lógico y diseño físico, como lo muestra la figura 1.2. El diseño de bases de datos, tal como se expone en este apartado, representa un enfoque orientado a los **datos** para el desarrollo de los sistemas de información: la atención completa del proceso de diseño se centra en los datos y sus propiedades. Con un enfoque orientado a los datos, primero se diseña la base de datos, luego las aplicaciones que la usan. Este método se desarrolló en la década de 1970, con el establecimiento de la tecnología de bases de datos.

Diseño conceptual. El diseño conceptual parte de la especificación de requerimientos y su resultado es el esquema conceptual de la base de datos. Un **esquema conceptual** es una descripción de alto nivel de la estructura de la base de datos, *independiente* del software de DBMS que se use para manipularla. Un **modelo conceptual** es un lenguaje que se usa para describir esquemas conceptuales. El propósito del diseño conceptual es describir el *contenido de información* de la base de datos, más que las *estructuras de almacenamiento* que se necesitarán para manejar esta información. En realidad, el diseño conceptual debe hacerse aun cuando la implantación final *no* use un DBMS, sino archivos convencionales y lenguajes de programación.

Diseño lógico. El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un **esquema lógico** es una descripción de la estructura de la base de datos que puede procesar el software de DBMS. Un **modelo lógico** es un lenguaje usado para especificar esquemas lógicos; los modelos lógicos más usados pertenecen a tres clases: **relacional**, **de redes** y **jerárquico**. El diseño lógico depende de la clase de modelo de datos usado por el DBMS, *no* del

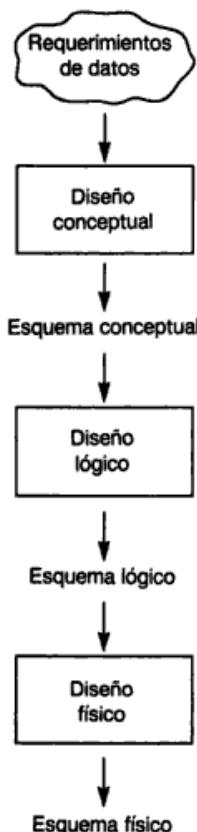


Figura 1.2. Enfoque orientado a los datos para el diseño de sistemas de información.

DBMS utilizado (en otras palabras, el diseño lógico se efectúa de la misma forma para todos los DBMS relativos porque todos utilizan el modelo relacional).

Diseño físico. El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implantación de una base de datos en la memoria secundaria; describe las estructuras de almacenamiento y los métodos usados para tener un acceso efectivo a los datos. Por esta razón, el diseño físico se adapta a un sistema DBMS específico. Hay una retroalimentación entre el diseño físico y el lógico, porque las decisiones tomadas durante el diseño físico para mejorar el rendimiento pueden afectar la estructura del esquema lógico.

Una vez completo el diseño físico de una base de datos, los esquemas lógico y físico se expresan haciendo uso del *lenguaje de definición de datos* del DBMS objetivo; la base de datos se crea y se carga, y puede ser probada. Más que eso,

Dependencia del:	El tipo de DBMS	Un DBMS específico
Diseño conceptual	No	No
Diseño lógico	Sí	No
Diseño físico	Sí	Sí

Figura 1.3. Dependencia de los diseños conceptual, lógico y físico de la clase de DBMS y el DBMS específico.

las aplicaciones que usan las bases de datos pueden especificarse, implantarse y probarse completamente. De este modo, la base de datos se vuelve paulatinamente operacional. La figura 1.3 resume la dependencia de los diseños conceptual, lógico y físico de la clase o tipo de DBMS y del DBMS específico.

1.3. Interacción entre el diseño de bases de datos y el análisis funcional

Un enfoque alternativo en el diseño de los sistemas de información, llamado **enfoque orientado a las funciones**, se muestra en la figura 1.4. Este enfoque se desarrolló en la década de 1960 y aún es muy popular. Difiere del enfoque orientado a los datos en que la atención se centra en las aplicaciones y no en los datos.

El análisis funcional arranca con los **requerimientos de aplicaciones**, descripciones de alto nivel de las actividades desarrolladas dentro de una organización y de los flujos de información intercambiados entre actividades. El resultado del análisis funcional es un conjunto de **esquemas de funciones**, que describen esas actividades y flujos de información mediante el uso de **modelos de funciones** específicos. En el análisis funcional, las bases de datos se consideran *depósitos de información* aislados, utilizados por cada actividad o intercambiados entre ellas; se pierde la visión de los datos como recurso *global* de la empresa.

La siguiente fase del diseño funcional, llamada **diseño de aplicaciones de alto nivel**, transforma los esquemas de función en **especificaciones de aplicación**, que describen, a un alto nivel de abstracción, la conducta de los programas de aplicación; en particular, describen cómo las aplicaciones obtienen acceso a las bases de datos. Estas especificaciones son la base para el **diseño de programas de aplicación posterior**, que produce una especificación detallada del programa de aplicación y, en última instancia, el código del programa.

En realidad, los enfoques orientados a los datos y a las funciones para el diseño de sistemas de información son complementarios; ambos aportan carac-

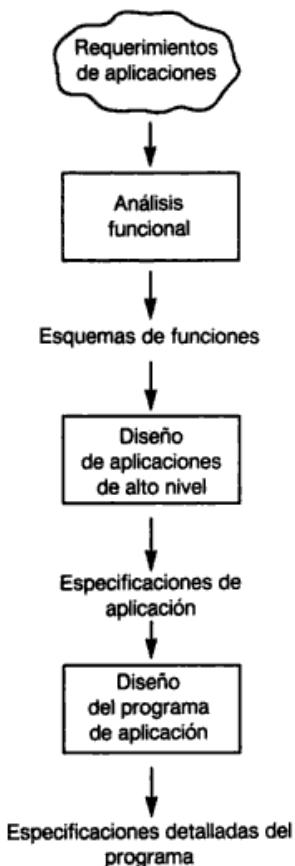


Figura 1.4. Enfoque orientado a las funciones para el diseño de sistemas de información.

terísticas positivas y se deben relacionar íntimamente. De este modo, si bien su enfoque primordial está orientado a los datos, este libro cubre ambos, presentando un enfoque conjunto **orientado a los datos y a las funciones** para el diseño de sistemas de información, como lo muestra la figura 1.5. Aclarado el objetivo de este libro, centraremos nuestra atención en la producción de los esquemas de bases de datos y nos desentenderemos de la producción de especificaciones detalladas para los programas de aplicación.

La idea básica de la metodología conjunta es crear el esquema conceptual de la base de datos y el esquema de funciones, paralelamente, de forma que los dos procesos de diseño se influyan mutuamente. En particular, la metodología conjunta hace posible comprobar que los esquemas de datos y de funciones sean mutuamente coherentes (es decir, que no generen conflicto) y completos (es de-

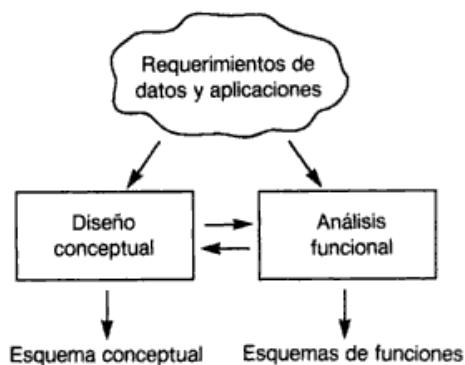


Figura 1.5. Enfoque conjunto orientado a los datos y las funciones para el diseño de sistemas de información.

cir, que todos los datos requeridos por las funciones se representen en el esquema conceptual de la base de datos y, a la inversa, que las funciones incluyan todas las operaciones requeridas por la base de datos).

1.4. Modelos y herramientas para el diseño de bases de datos y el análisis funcional

El diseño de bases de datos y el análisis funcional están fuertemente influidos por la elección de modelos adecuados para representar los datos y las funciones. Estos modelos, como los lenguajes de programación, poseen un conjunto fijo de construcciones lingüísticas que se pueden usar en la descripción de datos y funciones.

Es muy importante hacer notar que las construcciones de un modelo también cuentan con una *representación gráfica*, la que permite al diseñador crear diagramas y dibujos. Estos documentos son fáciles de leer y entender y son por ello ingredientes esenciales en el proceso de diseño.

A fines de los años setenta, varios modelos conceptuales de bases de datos se propusieron como alternativas; entre otros, el *modelo semántico de los datos*, el *modelo estructural*, el *modelo funcional*, varios tipos de *modelos binarios*, etcétera. De hecho, todos los modelos conceptuales se basan en el uso de unos cuantos mecanismos de abstracción y, en consecuencia, casi siempre es posible definir las correspondencias entre ellos. Particularmente, el **modelo de entidades-interrelaciones** surgió como la estructura formal más destacada para la representación conceptual de datos, llegando a imponerse como un estándar industrial. El modelo de entidades-interrelaciones se basa en sólo unos pocos

conceptos de modelado y posee una eficaz representación gráfica, en la que cada elemento del modelo corresponde a un símbolo gráfico distinto.

De forma similar, varios modelos fueron propuestos para el análisis funcional en los años setenta. Estos modelos son menos homogéneos que los modelos de datos y no tan fácilmente comparables, porque el análisis funcional se aplica a problemáticas muy distintas, que van desde el procesamiento convencional de datos hasta la planificación y control en tiempo real. En este libro, se ha hecho hincapié en las aplicaciones de procesamiento de datos y a partir de este contexto surgió el **modelo de flujo de datos**, transformándose en un estándar de la industria. Este modelo es simple y conciso, además, cada elemento del modelo corresponde a un símbolo gráfico distinto.

Recientemente, la atención se ha desplazado de los modelos de datos a las metodologías y herramientas de diseño. Se puede argumentar que un correcto enfoque metodológico puede ser, como mínimo, tan importante como la elección de los modelos de datos o funciones. Además, se ha desarrollado una variedad de herramientas de diseño asistidas por computador, muchas de las cuales apoyan una representación gráfica de los esquemas de datos y funciones. Las capacidades gráficas incluyen la edición mediante dispositivos apuntadores (ratones), visualización selectiva de partes del esquema y uso de ventanas múltiples para el seguimiento de diferentes aspectos del proceso de diseño. Las herramientas de diseño por lo regular apoyan el modelo de entidades-interrelaciones para datos y el modelo de flujo de datos para funciones.

1.5. Por qué es valioso el diseño conceptual

Al ser este libro partidario de dar importancia a un enfoque conceptual para el diseño de bases de datos, presenta una breve defensa de este enfoque. Primero, se debe subrayar que el diseño conceptual no se ayuda mucho de herramientas automáticas. El diseñador asume total responsabilidad sobre el proceso de entender y transformar los requerimientos en esquemas conceptuales. A partir de la primera conceptualización, muchos sistemas de bases de datos ofrecen herramientas para la creación rápida de prototipos, usando lenguajes de cuarta generación para la generación de aplicaciones, formatos de pantallas e informes. Estas herramientas pueden ser usadas directamente por no profesionales para desarrollar bases de datos simples y facilitan el trabajo de los creadores profesionales de bases de datos. Así pues, creemos que el diseño conceptual es con mucho la fase más crucial del diseño de bases de datos, y el desarrollo posterior de la tecnología de bases de datos no cambiará esta situación.

Aun si se supone que el diseño conceptual está dirigido por un profesional, se alcanzan resultados satisfactorios sólo mediante cooperación con los usuarios de bases de datos, quienes tienen la obligación de describir las necesidades y de explicar el significado de los datos. Las características básicas del diseño conceptual y de los modelos conceptuales de datos son relativamente simples y su en-

tendimiento no requiere mayor conocimiento técnico previo sobre sistemas de bases de datos. De este modo creemos que los usuarios pueden aprender fácilmente lo suficiente sobre diseño conceptual para orientar a los diseñadores en sus decisiones e incluso para diseñar bases de datos simples.

Una influencia fuerte del usuario final sobre las decisiones de diseño tiene muchas consecuencias positivas: mejora la calidad del esquema conceptual, eleva la probabilidad de que el proyecto converja hacia el resultado esperado, y reduce los costos de desarrollo. Más importante, los usuarios que estén más comprometidos con el proceso de toma de decisiones son los más inclinados a aceptar y usar un sistema de información. Entender las características de las bases de datos mejora la claridad contractual entre los grupos implicados, es decir, usuarios y diseñadores.

Otro argumento de fuerza a favor del diseño conceptual es su independencia de un DBMS particular. Esta característica genera algunas ventajas:

1. La elección del DBMS se puede posponer, y el esquema conceptual puede sobrevivir a una decisión tardía de cambiar el DBMS objetivo.
2. Si el DBMS o los requerimientos de la aplicación cambian, el esquema conceptual puede aún usarse como punto de partida de la nueva actividad de diseño.
3. Las diferentes bases de datos, descritas mediante su esquema conceptual, se pueden comparar en un marco homogéneo de trabajo. Esta característica facilita la construcción de sistemas consolidados a partir de varias bases de datos ya existentes y la creación de un diccionario de datos integrado.

El argumento final en favor del diseño conceptual subraya la utilidad de los esquemas conceptuales después de finalizado el proceso de diseño. El esquema conceptual no se debe considerar un documento de diseño intermedio a ser ignorado después del diseño lógico y físico; más bien, debe permanecer como parte de las especificaciones de la base de datos, organizado con una variedad de documentos que también detallan la recolección de requerimientos y el proceso de diseño. Así, la ventaja final y posiblemente más importante del diseño conceptual se hace evidente durante la operación de la base de datos, cuando el modelo conceptual y su documentación facilitan el entendimiento de los esquemas de datos y de las aplicaciones que los utilizan y, por ende, su transformación y mantenimiento.

1.6. Resumen

En este capítulo se ha presentado un enfoque global para el diseño de bases de datos. Se ha colocado al diseño de bases de datos dentro de una amplia perspectiva de ingeniería de software al mostrar su papel dentro del ciclo de vida de los sistemas de información. Luego se ha presentado una visión interna del proceso

al descomponer el diseño de datos en tres fases secuenciales de diseño conceptual, lógico y físico. Finalmente, se ha considerado la relación entre el diseño de bases de datos y el análisis funcional al introducir las nociones de los enfoques orientado a los datos y orientado a las funciones, y al presentar un enfoque conjunto orientado a los datos y a las funciones.

Se ha indicado que dos modelos, el de entidades-interrelaciones y el de flujo de datos, han surgido como estándares industriales para el diseño conceptual de bases de datos y el análisis funcional. Comparten ciertas propiedades, incluidas legibilidad, sencillez y una eficaz representación gráfica. La mayoría de las herramientas de apoyo al diseño de sistemas de información se basan en estos dos modelos. Se han resaltado tanto las dificultades del diseño conceptual de bases de datos como las ventajas del uso adecuado de este enfoque, que incluyen la motivada participación del usuario en el proceso de diseño, la independencia del DBMS y un mejor mantenimiento de esquemas y aplicaciones en el largo plazo.

Bibliografía

W. Davis, *System Analysis and Design: A Structured Approach*, Addison-Wesley, 1983.

R. Fairley, *Software Engineering Concepts*, McGraw-Hill, 1985.

C. Gane y T. Sarson, *Structured System Analysis: Tools and Techniques*, Prentice-Hall, 1979.

Estos libros abordan los aspectos generales del diseño de sistemas de información, desde la definición del problema inicial y estudio de factibilidad hasta la fase de implantación. Davis (1983) proporciona la introducción más completa en esta área. Gane y Sarson (1979) tratan principalmente la fase de análisis funcional, y Fairley (1985) muestra varias relaciones entre el análisis funcional y el diseño de software.

A. Cárdenas, *Data Base Management Systems*, 2.^a ed., Allyn and Bacon, 1985.

C. J. Date, *An Introduction to Database Systems*, 5.^a ed., Addison-Wesley, 1990*.

C. J. Date, *An Introduction to Database Systems*, vol. 2, Addison-Wesley, 1983.

R. Elmasri y S. B. Navathe, *Fundamentals of Database Systems*, Benjamin/Cummings, 1989.

H. Korth y A. Silberschatz, *Database Systems Concepts*, McGraw-Hill, 1986.

F. R. McFadden y J. A. Hoffer, *Data Base Management*, Benjamin/Cummings, 1985.

J. Martin, *Computer Data-Base Organization*, 2.^a ed., Prentice-Hall, 1975.

J. Ullman, *Principles of Data and Knowledge Based Systems*, Computer Science Press, 1989.

Los textos anteriores son introducciones detalladas a los sistemas de bases de datos. Abarcan temas como arquitectura de bases de datos, modelos de datos, lenguajes de manipulación y de consulta, organización física de los datos, gestión de bases de datos, seguridad, manejo de transacciones, control de concurrencia, fiabilidad, teoría relacional y bases de datos distribuidas. Cárdenas (1985) tiene la más amplia introducción a los sistemas comerciales; Date (1990) está recomendado para usuarios de DB2 y de sistemas y lenguajes relacionales, McFadden y Hoffer (1985) para cuestiones de planificación y ad-

* Existe versión en castellano publicada por Addison-Wesley Iberoamericana, 1993. (N. del E.)

ministración de bases de datos, y Ullman (1989) para un tratamiento teórico riguroso de la teoría de bases de datos, enfocado sobre todo al modelo relacional. Elmasri y Navathe (1989) mantienen un punto de vista imparcial de los tres modelos de implantación (relacional, de redes y jerárquico). Su libro es particularmente apropiado para adquirir una buena base técnica con vistas a realizar diseño de bases de datos.

- D. R. Howe, *Data Analysis for Data Base Design*, E. Arnold, 1983.
M. E. S. Loomis, *The Database Book*, Macmillan, 1987.
T. Teorey y J. Fry, *Design of Database Structures*, Prentice-Hall, 1982.
G. Wiederhold, *Database Design*, 2.^a ed., McGraw-Hill, 1984.
G. Wiederhold, *Database Design for File Organizations*, McGraw-Hill, 1987.

Estos libros se ocupan del diseño de bases de datos. Howe (1983) está orientado principalmente al diseño lógico y conceptual, mientras que Teorey y Fry (1982) y Wiederhold (1984) tratan exhaustivamente el diseño físico. Wiederhold (1987) presenta una serie de técnicas para el diseño de organizaciones de sistemas de archivos. Loomis (1987) usa una técnica semántica para el modelado de datos y muestra cómo los modelos conceptuales pueden servir como fundamento para el diseño de bases de datos relacionales, jerárquicas y de redes.

W. Kent, *Data and Reality*, North-Holland, 1978.

Una introducción estimulante a la naturaleza de la información. Quizá el mejor análisis de la función de los modelos de datos —su utilidad y límites— en la representación de la información.

S. Atre, *Structured Techniques for Design, Performance, and Management of Databases*, Wiley, 1980.

Referencia práctica y fácil de leer para el diseño de bases de datos en los tres modelos; da opciones de diseño y orientaciones para la elección entre ellos.

W. H. Inmon, *Effective Database Design*, Prentice Hall, 1981.

Introducción general y elemental a los problemas del diseño de bases de datos y aplicaciones.

J. L. Weldon, *Database Administration*, Plenum Press, 1981.

Una de las pocas referencias dedicada a la gestión de bases de datos.

M. L. Brodie y S. N. Zilles, eds., «Workshop on Data Abstraction, Database and Conceptual Modeling», *ACM SIGMOD Record* 11, núm. 2, 1981.

V. Lum et al., «1978 New Orleans Data Base Design Workshop Report», *Proc. Fifth International Conference on Very Large Data Bases*, Río de Janeiro, 1979.

S. B. Navathe y L. Kerschberg, «Role of Data Dictionaries in Information Resource Management», *Information and Management* 10, núm. 1 (1986), 21-48.

S. B. Yao, S. B. Navathe, T. Kunii y J. L. Weldon, eds., «Database Design Techniques, 1: Requirements and Logical Structures», *Proceedings of the NYU Symposium on Logical Database Design, Lecture Notes in Computer Science* 132, 1982, Springer-Verlag.

Estas cuatro publicaciones tuvieron una influencia importante en la definición de la estructura general de una metodología para el diseño de bases de datos. Diferencian las cuatro fases, análisis de requerimientos, diseño conceptual, diseño lógico y diseño físico, e indican las cualidades de las metodologías y modelos de datos.

Conceptos sobre el modelado de datos

Los modelos de datos son vehículos para describir la realidad. Los programadores usan los modelos de datos para construir esquemas, los cuales son representaciones de la realidad. La calidad de los esquemas resultantes depende no sólo de la habilidad de los programadores, sino también de las características del modelo de datos seleccionado.

El bloque de construcción común a todos los modelos de datos es una pequeña colección de «mecanismos de abstracción» primitivos: *clasificación, agregación y generalización*. Las abstracciones ayudan al programador a entender, clasificar y modelar la realidad; se describen en el apartado 2.1.

Mediante las abstracciones, el programador es capaz de clasificar los objetos del mundo real y modelar las interrelaciones de las distintas clases. En particular, las abstracciones de agregación y generalización establecen correspondencias complejas entre las clases; el apartado 2.2 describe las propiedades de estas correspondencias.

El apartado 2.3 define los modelos de datos, esquemas y casos de bases de datos. Asimismo, señala la diferencia entre modelos conceptuales y lógicos y expone las características de los modelos conceptuales y de su representación gráfica.

El apartado 2.4 describe el modelo conceptual usado en este libro, el modelo de entidades-interrelaciones (ER). El modelo ER es la herramienta fundamental para el diseñador de bases de datos. Como es necesario estar bien informado sobre las entidades, las interrelaciones y otros conceptos del modelo ER, se sugiere no omitir este apartado, aun conociendo el modelo ER. Finalmente, el apartado 2.5 enseña cómo leer un esquema ER.

2.1. Abstracciones en el diseño conceptual de bases de datos

La **abstracción** es un proceso mental que se aplica al seleccionar algunas características y propiedades de un conjunto de objetos y excluir otras no pertinentes.

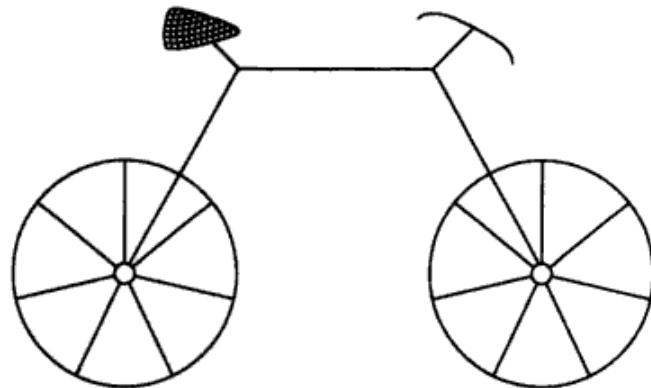


Figura 2.1. La abstracción *bicicleta*.

En otras palabras, se hace una abstracción al fijar la atención en las propiedades consideradas esenciales de un conjunto de cosas y desechar sus diferencias.

En la figura 2.1 vemos una bicicleta; el concepto de *bicicleta* puede verse como resultado de un proceso de abstracción, lo que hace excluir todos los detalles de la estructura de una bicicleta (la cadena, los pedales, los frenos, etc.) y todas las posibles diferencias entre bicicletas. Comúnmente se asocia un nombre con cada abstracción. El dibujo de la bicicleta de la figura 2.1 es una representación de esta abstracción. Otra representación sería una descripción en castellano del mismo dibujo.

En el diseño conceptual de bases de datos se usan tres tipos de abstracciones: *clasificación*, *agregación* y *generalización*. Los apartados siguientes presentan cada una de ellas.

2.1.1. Abstracción de clasificación

La **abstracción de clasificación** se usa para definir un concepto como una *clase* de objetos de la realidad caracterizados por propiedades comunes. Por ejemplo, tenemos que el concepto *bicicleta* es la clase cuyos miembros son todos bicicletas (la bicicleta roja, la bicicleta de Tomás, etc.); asimismo, el concepto *mes* es la clase cuyos miembros son enero, febrero, ..., diciembre. Cuando se piensa en un *mes* (por ejemplo, los pagos por el alquiler de una casa se harán cada mes) se hace una abstracción de las características específicas de cada mes (por ejemplo, número de días) y se destacan los aspectos comunes de todos los meses: son grupos de días con límites bien definidos (el primer y último día) y con dimensiones aproximadamente iguales (28 a 31 días).

Se representa gráficamente la clasificación como un árbol de un nivel, que tiene como raíz la clase y como hojas los elementos de la clase (Fig. 2.2); las ramas del árbol se representan por líneas discontinuas. Cada rama del árbol in-



Figura 2.2. Un ejemplo de clasificación.

dica que un nodo hoja es un miembro de (*ES_MIEMBRO_DE*) la clase que representa la raíz.

Un mismo objeto real puede clasificarse de varias maneras. Por ejemplo, consideremos los siguientes grupos de objetos:

{silla negra, mesa negra, silla blanca, mesa blanca}

Se puede clasificar los objetos anteriores como MESAS y SILLAS o considerar, en cambio, su color y clasificarlos como MUEBLES NEGROS y MUEBLES BLANCOS. Claramente, se obtienen clases con diferentes elementos, como lo muestra la figura 2.3. Este ejemplo muestra también que cada objeto real puede pertenecer a varias clases.

2.1.2. Abstracción de agregación

Una **abstracción de agregación** define una nueva clase a partir de un conjunto de (otras) clases que representan sus partes componentes. Se aplica esta abstracción cuando, partiendo de las clases RUEDA, PEDAL, MANILLAR, etc., se forma la clase BICICLETA. De igual forma, se aplica una **agregación** cuando se abstrae la clase PERSONAS partiendo de las clases NOMBRE, SEXO y SALARIO. La abstracción por agregación se representa por un árbol de un nivel en el cual todos los nodos son clases; la raíz representa la clase creada por agregación de las clases representadas por las hojas. Cada rama del árbol indica que una clase hoja es una parte de (*ES_PARTE_DE*) la clase representada por la raíz. Para distinguir la *agregación de la clasificación*, las ramas dirigidas están representadas por líneas dobles que van de los componentes a los objetos agregados (Fig. 2.4).

En el ejemplo de la clase PERSONAS, un elemento de la clase raíz corresponde exactamente a un elemento de las clases hojas (cada persona tiene un

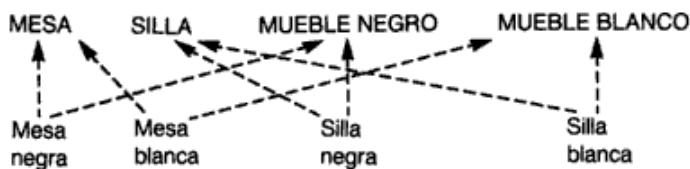


Figura 2.3. Clasificaciones múltiples para los mismos objetos de la realidad.

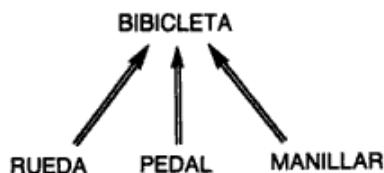


Figura 2.4. Ejemplo de agregación.

nombre, un sexo y un salario). Sin embargo, una forma más compleja de agregación se da cuando se considera las abstracciones USA como una agregación de PERSONA y EDIFICIO. Aquí, en general, una persona hace uso de múltiples edificios y un edificio es habitado o usado por varias personas. En el apartado siguiente, se habla sobre las propiedades de las agregaciones que establecen correspondencias complejas entre las clases.

La clasificación y la agregación son las dos abstracciones básicas utilizadas para construir estructuras de datos dentro de las bases de datos y de muchos lenguajes convencionales de programación. La clasificación es el procedimiento utilizado cuando, partiendo de elementos individuales de información, se identifican *tipos de campos* o *atributos*. La agregación es el procedimiento mediante el cual se reúnen tipos de campos relacionados en grupos como por ejemplo *tipos de registros* (Fig. 2.5). Otra vez, la clasificación es la abstracción que se hace al asignar varios registros (ocurrencias) de estudiantes, por ejemplo, a un tipo de registro llamado tipo de registro ESTUDIANTE.

2.1.3. Abstracción de generalización

Una **abstracción de generalización** define una relación de subconjunto entre los elementos de dos (o más) clases. Por ejemplo, la clase VEHICULO es una generalización de la clase BICICLETA, ya que todas las bicicletas son vehículos.

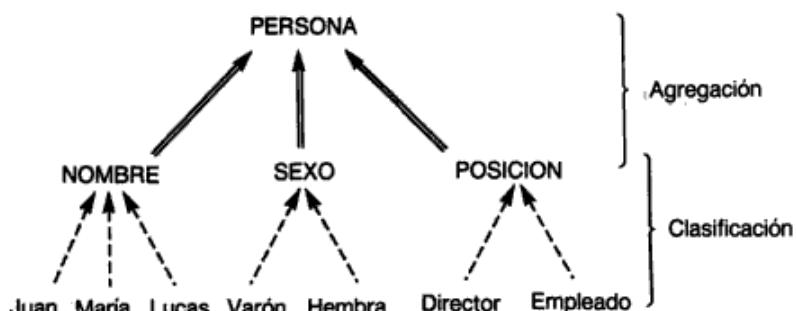


Figura 2.5. Clasificación y agregación como mecanismos básicos para la creación de estructuras convencionales de datos.

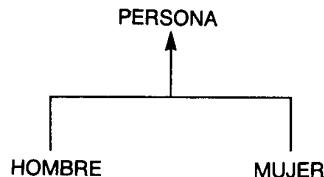


Figura 2.6. Un ejemplo de generalización.

Asimismo, la clase PERSONA es una generalización de las clases HOMBRE y MUJER (Fig. 2.6). Cada generalización se representa con un árbol de un nivel, en el que todos los nodos son clases, con la clase genérica como raíz y las clases subconjunto como hojas; cada rama del árbol expresa que una clase hoja es un (ES_UN) subconjunto de la clase raíz. Para distinguir la generalización de otras abstracciones, se usa una flecha sencilla apuntando hacia la raíz. La abstracción de generalización, a pesar de ser muy común e intuitiva, no se usa en muchos modelos de datos. Sin embargo, es muy útil por su cualidad fundamental de herencia: *en una generalización, todas las abstracciones definidas para la clase genérica son heredadas por las clases subconjunto*.

Consideremos una vez más las figuras 2.5 y 2.6. En la primera, PERSONA es un agregado de NOMBRE, SEXO y POSICION. En la segunda, HOMBRE y MUJER son subconjuntos de PERSONA. La propiedad de herencia permite deducir que HOMBRE también puede ser considerado como un agregado de NOMBRE, SEXO y POSICION. Sin embargo, los hombres también pueden caracterizarse por rasgos adicionales (como su SITUACION_MILITAR) que no son necesariamente comunes a todas las personas. Este proceso de herencia se muestra en la figura 2.7. Así, las generalizaciones y la herencia permiten a los diseñadores construir estructuras compactas de conceptos. Nótese que, en la representación gráfica, las flechas indican la dirección de la abstracción, señalando hacia el concepto más abstracto. En la figura 2.7, las flechas dobles representan una agregación (PER-

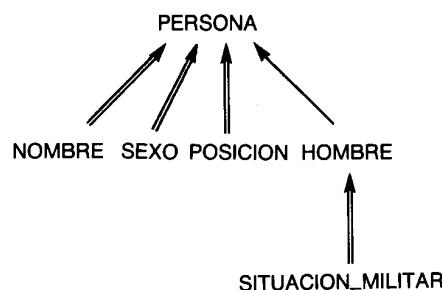


Figura 2.7. Herencia con abstracciones de generalización.



Figura 2.8. Las agregaciones binarias USA y POSEE.

SONA es una agregación de NOMBRE, SEXO y POSICION), mientras que la flecha sencilla de HOMBRE a PERSONA indica una generalización o una relación de subconjunto.

Las tres abstracciones son independientes: ninguna de ellas puede describirse en función de las otras, y cada una de ellas proporciona un mecanismo diferenciado en el proceso de estructuración de la información. La independencia de las abstracciones se hace evidente al razonar sobre las propiedades matemáticas de las tres interrelaciones de conceptos establecidas por las abstracciones: la clasificación corresponde a la membresía (interrelación de pertenencia) de conjuntos (ES_MIEMBRO_DE); la agregación, a la composición de conjuntos (ES_PARTE_DE); y la generalización, a la inclusión (interrelación de subconjunto) en conjuntos (ES_UN).

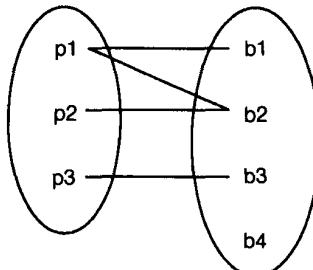
2.2. Propiedades de las correspondencias entre las clases

Las abstracciones de agregación y generalización establecen correspondencias entre clases. En este apartado se estudian las características de estas correspondencias. Primero se consideran las agregaciones binarias, que son agregaciones entre dos clases; después se consideran las agregaciones n-arias o agregaciones entre tres o más clases. Por último, se consideran las generalizaciones.

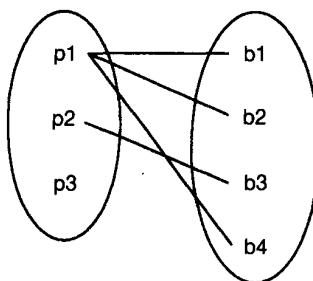
2.2.1. Agregación binaria

Una **agregación binaria** es una correspondencia que se establece entre dos clases. Por ejemplo, USA es una agregación binaria de las clases PERSONA y EDIFICIO, que establece una *correspondencia* entre los elementos de las dos clases; claramente, se interpreta esta correspondencia como el hecho de que una persona determinada utiliza un edificio determinado. Asimismo, CONDUCE es una agregación binaria de las clases PERSONA y COCHE, con una clara interpretación.

Podemos establecer varias agregaciones binarias entre dos clases; por ejemplo, podemos agregar PERSONA y CASA para formar un nuevo concepto POSEE establecido entre una casa y su propietario. La figura 2.8 muestra las dos agregaciones binarias USA y POSEE, para las dos clases PERSONA y EDIFICIO.



(a) Agregación binaria USA



(b) Agregación binaria POSEE

Figura 2.9. Representación de las agregaciones binarias USA y POSEE entre PERSONA y EDIFICIO.

Se puede representar una **correspondencia binaria** entre dos clases como sigue: se describe a las dos clases como conjuntos y se traza una línea desde un elemento de un conjunto a un elemento del otro conjunto cada vez que los dos elementos estén agregados. La figura 2.9a muestra la correspondencia binaria USA entre PERSONA y EDIFICIO; la figura 2.9b muestra la correspondencia binaria POSEE entre PERSONA y EDIFICIO. Al comparar las dos figuras, se nota que las correspondencias se caracterizan por propiedades diferentes; por ejemplo, cada persona habita un edificio, pero pocas personas poseen un edificio. Así mismo, cada edificio puede ser usado por varias personas pero pertenecer a una sola persona. Estas observaciones se refieren a la *cardinalidad* de la correspondencia entre las clases y se describen en detalle en la exposición que sigue.

Cardinalidad mínima (card-mín). Considérese una agregación A entre las clases C_1 y C_2 . La cardinalidad mínima de C_1 en A, denotada por card-mín (C_1, A), es el menor número de correspondencias en las que cada elemento de C_1 puede tomar parte. Asimismo, la cardinalidad mínima de C_2 en A, represen-

tada como card-mín (C_2, A), es el menor número de correspondencias en las que cada elemento de C_2 puede participar.

Considérense las agregaciones USA y POSEE entre PERSONA y EDIFICIO:

1. Si se supone que cada persona usa al menos un edificio, entonces, card-mín (PERSONA,USA) = 1.
2. Si se supone que algunos edificios no están habitados, entonces, card-mín (EDIFICIO,USA) = 0.
3. Si se supone que algunas personas no poseen un edificio, entonces, card-mín (PERSONA,POSEE) = 0.
4. Si se supone que cada uno de los edificios debe pertenecer a una persona, entonces, card-mín (EDIFICIO,POSEE) = 1.

Estos ejemplos muestran que los valores importantes de card-mín (C_1, A) son 0 y 1, respectivamente. La cardinalidad mínima raras veces adopta valores diferentes; sin embargo, son posibles valores mayores. Por ejemplo, si consideramos la correspondencia binaria OBLIGACION_SEMANAL entre PERSONA y DIAS_DE_LA_SEMANA, y suponemos que cada persona tiene al menos una obligación por día (con 5 días laborables a la semana), entonces, card-mín (PERSONA,OBLIGACION_SEMANAL) = 5.

Si card-mín (C_1, A) = 0, se dice que la clase C_1 tiene una **participación opcional** en la agregación, porque algunos elementos de la clase C_1 pueden *no* tener correspondencia en la agregación A con elementos de la clase C_2 . Si card-mín (C_1, A) > 0, se dice que la clase C_1 tiene una **participación obligatoria** en la agregación, ya que cada elemento de la clase C_1 debe corresponder, al menos, a un elemento de la clase C_2 . En los ejemplos antes citados, la participación de la entidad EDIFICIO en la relación USA es opcional; la participación de la entidad EDIFICIO en la relación POSEE es obligatoria.

Cardinalidad máxima (card-máx). Considérese la agregación A entre las clases C_1 y C_2 . La cardinalidad máxima de C_1 en A, que se representa como card-máx (C_1, A), es el mayor número de correspondencias en las que cada elemento de C_1 puede participar. Asimismo, la cardinalidad máxima de C_2 en A, denotada por card-máx (C_2, A), es el mayor número de correspondencias en las que puede participar cada elemento de C_2 .

Consideremos de nuevo las agregaciones USA y POSEE entre PERSONA y EDIFICIO:

1. Si se supone que cada persona usa varios edificios, card-máx (PERSONA,USA) = n. Por n se entiende «infinito»¹ o «ilimitado».
2. Si se supone que cada EDIFICIO puede tener varios habitantes, entonces, card-máx (EDIFICIO,USA) = n.

¹ El número estará en realidad limitado al número de ocurrencias de la clase EDIFICIO en la base de dato.

3. Si se supone que cada persona puede poseer varios edificios, entonces, card-máx (PERSONA,POSEE) = n.
4. Si se supone que cada edificio pertenece sólo a una persona, entonces, card-mín (EDIFICIO,POSEE) = 1 y card-máx (EDIFICIO,POSEE).

Estos ejemplos muestran que los valores importantes para card-máx (C_1, A) son 1 y n; n representa cualquier número e indica que cada elemento de C_1 puede pertenecer a un número arbitrariamente grande de correspondencias. Card-máx pocas veces adopta un valor fijo, pero puede darse el caso. Por ejemplo, si se considera la correspondencia binaria FIESTAS_OFICIALES entre PERSONA y DIAS_DE_LA_SEMANA, y se supone que cada persona cuenta con no más de dos días de fiesta oficiales por semana, entonces, card-máx (PERSONA,FIESTAS_OFICIALES) = 2.

Si card-máx (C_1, A) = 1 y card-máx (C_2, A) = 1, se dice que la agregación es de **uno a uno**. Si card-máx (C_1, A) = n y card-máx (C_2, A) = 1, la agregación de C_2 a C_1 es de **uno a muchos**. Si card-máx (C_1, A) = 1 y card-máx (C_1, A) = n, la agregación de C_1 a C_2 es de **muchos a uno**; por último, si card-máx (C_1, A) = m y card-máx (C_2, A) = n (donde m y n representan valores superiores a 1), la agregación es de **muchos a muchos**.

Estos conceptos se aplican ampliamente en el diseño de bases de datos, a veces de manera incorrecta. La figura 2.10 muestra un ejemplo para cada uno de los cuatro tipos de correspondencias entre las clases C_1 y C_2 . Se caracteriza por completo cada participación de una clase en una agregación al indicar los dos valores de mínima y máxima cardinalidad. Esto se representa como sigue: sea A una agregación binaria de las clases C_1 y C_2 , con card-mín (C_1, A) = m_1 y card-máx (C_1, A) = M_1 ; entonces, se dice que la **cardinalidad** de C_1 en A es el par (m_1, M_1): card (C_1, A) = (m_1, M_1). De igual modo, si card-mín (C_2, A) = m_2 y card-máx (C_2, A) = M_2 , card (C_2, A) = (m_2, M_2). Estos conceptos serán muy útiles en el análisis que se hace más adelante sobre las características de las interrelaciones en el modelo ER.

2.2.2. Agregación n-aria

Una **agregación n-aria** es una correspondencia establecida entre tres o más clases. Por ejemplo, SE_IMPARTE es una agregación ternaria entre las clases CURSO, DIA y AULA. Expresa la idea de que un determinado curso se imparte un cierto día en un aula determinada. Igual que ocurre con las relaciones binarias, es importante describir las propiedades de cardinalidad de esta correspondencia. Resulta que las cardinalidades mínima y máxima se definen exactamente de la misma manera.

Cardinalidad mínima (card-mín). Considérese la agregación A entre las clases C_1, C_2, \dots, C_n . La cardinalidad mínima de C_i en A, representada por card-mín (C_i, A), es el número mínimo de correspondencias en las que cada elemento de C_i puede participar.

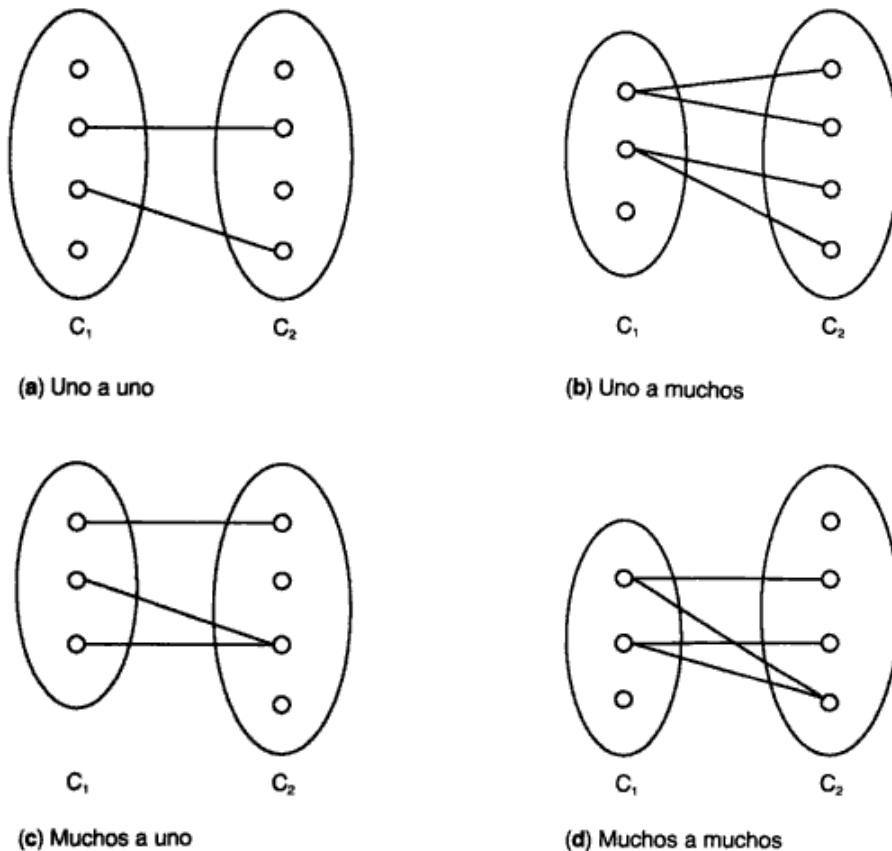


Figura 2.10. Ejemplos de correspondencias de uno a uno, uno a muchos, muchos a uno y muchos a muchos.

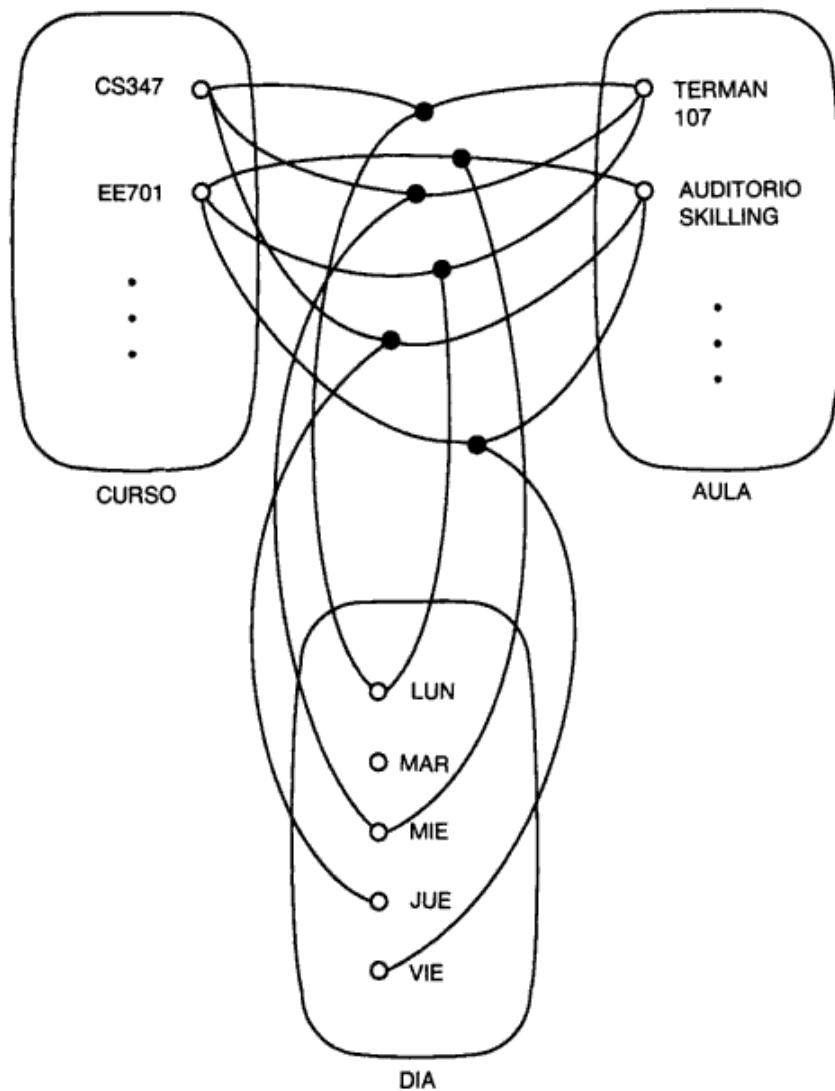
Cardinalidad máxima (card-máx). Considérese la agregación A entre las clases C_1, C_2, \dots, C_n . La cardinalidad máxima de C_i en A, denotada por $\text{card-máx}(C_i, A)$, es el número máximo de correspondencias en las que cada elemento de C_i puede participar.

Considérese la agregación ternaria SE_IMPARTE entre las clases curso, día y aula, representada en la figura 2.11.

1. Si se supone que cada curso puede impartirse entre una y tres veces por semana, entonces

$$\begin{aligned}\text{card-mín}(\text{CURSO}, \text{SE_IMPARTE}) &= 1 \\ \text{card-máx}(\text{CURSO}, \text{SE_IMPARTE}) &= 3\end{aligned}$$

2. Si se supone que en cada día de la semana puede impartirse cualquier número de cursos, entonces

(a) SE_IMPARTE como una agregación ternaria(b) Correspondencias en la agregación SE_IMPARTE**Figura 2.11.** Representación de la agregación ternaria SEIMPARTE.

card-mín (DIA,SE_IMPARTE) = 0
 card-máx (DIA,SE_IMPARTE) = n^2

3. Si se supone que en cada aula puede haber como máximo cuarenta reuniones por semana (ocho reuniones por día, cinco días por semana), entonces

card-mín (AULA,SE_IMPARTE) = 0
 card-máx (AULA,SE_IMPARTE) = 40

Como en el caso de las agregaciones binarias, se caracteriza por completo cada participación de una clase en una agregación al indicar los dos valores de la mínima y máxima cardinalidad. Por consiguiente:

card (CURSO,SE_IMPARTE) = (1,3)
 card (DIA,SE_IMPARTE) = (0,n)
 card (AULA,SE_IMPARTE) = (0,40)

2.2.3. Generalizaciones

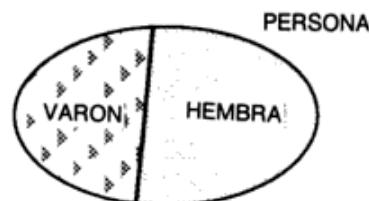
Una **abstracción de generalización** establece una correspondencia entre la clase genérica (raíz) y las clases subconjunto. Considérese la clase PERSONA como una generalización de las clases VARON y HEMBRA; cada elemento de éstas corresponde exactamente a un elemento de la clase PERSONA. En esta generalización, cada persona corresponde también a un elemento de la clase VARON o a un elemento de la clase HEMBRA; sin embargo, esto no ocurre en todas las generalizaciones. Estas observaciones se refieren a las **propiedades de cobertura** de la generalización, que se describen formalmente a continuación.

Cobertura total o parcial. La cobertura de una generalización es **total** (t) si cada elemento de la clase genérica corresponde *al menos* a un elemento de las clases subconjunto; es **parcial** (p) si existe algún elemento de la clase genérica que no corresponde a *ningún* elemento de las clases subconjunto.

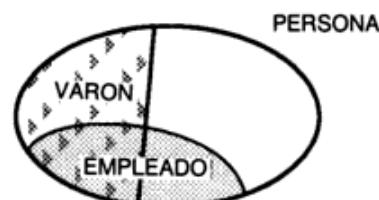
Cobertura exclusiva o superpuesta. La cobertura de una generalización es **exclusiva** (e) si cada elemento de la clase genérica corresponde, *a lo máximo*, a un elemento de las clases subconjunto; es **superpuesta** (s) si, al contrario, existe algún elemento de la clase genérica que corresponde a elementos de dos o más clases subconjunto diferentes. La figura 2.12 muestra todas las combinaciones de valores de cobertura representando ejemplos de clases como conjuntos e indicando cómo se superponen.

1. La cobertura de una generalización PERSONA de las clases VARON y HEMBRA es total y exclusiva: (t,e).

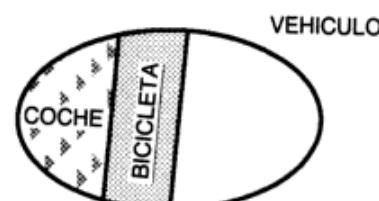
² El número aquí también está limitado; en este caso, a ocho veces el número de aulas.



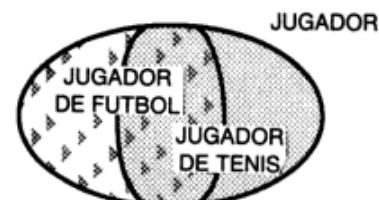
(a) Total, exclusiva



(b) Parcial, superpuesta



(c) Parcial, exclusiva



(d) Total, superpuesta

Figura 2.12. Valores de cobertura en las abstracciones de generalización.

2. La cobertura de una generalización PERSONA de las clases VARON y EMPLEADO es parcial y superpuesta: (p,s).
3. La cobertura de una generalización VEHICULO de las clases BICICLETA y COCHE es parcial y exclusiva: (p,e).

4. La cobertura de una generalización JUGADOR de las clases JUGADOR_DE_FUTBOL y JUGADOR_DE_TENIS, en un club que requiere que el socio pratique al menos dos de estos deportes, es total y superpuesta: (t,s).

Una generalización total y exclusiva corresponde a una *partición*, en el sentido matemático, de la clase genérica.

La anterior exposición sobre las abstracciones de datos y las propiedades de las correspondencias ha mostrado los instrumentos necesarios para el estudio de los modelos de datos y, en particular, del modelo de entidades-interrelaciones.

2.3. Modelos de datos

Un **modelo de datos** es una serie de conceptos que puede utilizarse para describir un conjunto de datos y operaciones para manipular los mismos. Cuando un modelo de datos describe un conjunto de conceptos de una realidad determinada, se llama **modelo conceptual de datos**. Los conceptos de un modelo de datos se construyen por lo regular usando mecanismos de abstracción y se describen mediante representaciones lingüísticas y gráficas; es decir, puede definirse una sintaxis y puede desarrollarse una notación gráfica como partes de un modelo de datos.

Hay dos tipos de modelos de datos: modelos conceptuales, usados en el diseño de bases de datos, y modelos lógicos, apoyados por los sistemas de gestión de bases de datos (DBMS), que son grandes paquetes de software que crean, modifican y mantienen bases de datos. Los **modelos conceptuales** son instrumentos para representar la realidad a un nivel alto de abstracción. Utilizando los modelos conceptuales, podemos construir una descripción de la realidad, fácil de entender e interpretar. Los **modelos lógicos** apoyan descripciones de datos procesables por un computador; incluyen el modelo *jerárquico*, el modelo CODASYL (*o de redes*) y el modelo *relacional*. Estos modelos tienen una correspondencia sencilla con la estructura física de la base de datos. En este capítulo se presentan los modelos conceptuales y en especial el modelo de entidades-interrelaciones. Los modelos lógicos se presentarán en la tercera parte de este libro.

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad; después se transforma el esquema conceptual en un esquema lógico. La razón de este enfoque radica en la dificultad de abstraer la estructura de bases de datos complejas. Un **esquema** es una representación de una parte específica de la realidad, creada usando un determinado modelo de datos. Dicho con mayor propiedad, un esquema es un conjunto estático de representaciones lingüísticas o gráficas, invariables en el tiempo, que describen la estructura de los datos de interés, como por ejemplo los de una organización.

Ilustraremos los conceptos de modelo y esquema construyendo un modelo de datos y un esquema de muestra, usando un subconjunto de las estructuras

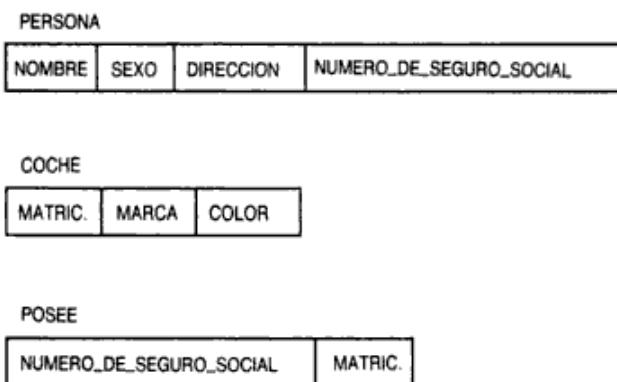


Figura 2.13. Esquema de muestra.

de datos que proporciona el lenguaje de programación Pascal. El modelo de datos de muestra contiene los conceptos de campo y registro. Cada *campo* del modelo de muestra puede pertenecer a uno de los siguientes tipos: entero, real, carácter o array³ de caracteres. Los enteros, caracteres y reales son tipos básicos de Pascal, mientras que los arrays son conjuntos de elementos del mismo tipo; se toma en consideración sólo arrays de caracteres. Cada *registro* en el modelo de datos de muestra es, simplemente, un conjunto de campos; éste es un concepto más simple que el de tipo de registro en Pascal, porque en ese lenguaje es posible definir registros de registros.

El esquema de muestra de la figura 2.13 describe una realidad en la que las personas poseen coches; abarca tres registros (PERSONA, POSEE, COCHE) y siete campos (NOMBRE, SEXO, DIRECCION, NUMERO_DE_SEGURO_SOCIAL, MATRÍCULA, MARCA, COLOR). Este esquema corresponde a dos clases, modeladas por los tipos de registro PERSONA y COCHE, y una agregación binaria entre las dos clases, modelada por el tipo de registro POSEE.

Cada *caso* de un esquema es una colección de datos dinámica, variable en el tiempo, que se ajusta a la estructura de datos que define el esquema. Cada esquema puede tener múltiples casos; el estado de la base de datos en un momento determinado corresponde a uno de esos casos. La evolución de la base de datos puede verse como la transición de un caso a otro, originada por alguna operación de modificación de datos. Se aplica el término *caso* también a un elemento del esquema para denotar la colección de datos que se refieren a ese elemento en particular. El contexto permite entender cuándo el término se refiere al esquema en general y cuándo a un solo elemento.

El caso del esquema de la figura 2.13, mostrado en la figura 2.14a, representa tres personas y cuatro coches. Cada coche pertenece a una persona. Nótese que la figura 2.14a representa un estado particular de la realidad; sin em-

³ En la comunidad latinoamericana, el término original «array» se traduce normalmente por «array». En el texto se mantiene el término original.

PERSONA

John Smith	M	11 West 12 St. Ft. Lauderdale	387-6713-362
Mary Smith	F	11 West 12 St. Ft. Lauderdale	389-4816-381
John Dole	M	1102 Ramona St. Palo Alto	391-3873-132

COCHE

CA 13718	Maserati	Blanco	387-6713-362	FL 18MIAI
FL 18MIAI	Porsche	Azul	387-6713-362	FL 176854
CA CATA17	Datsun	Blanco	391-3873-132	CA 13718
FL 171899	Ford	Rojo	391-3873-132	CA CATA17

(a) Caso de muestra

COCHE

CA 13718	Maserati	Blanco	387-6713-362	FL 18MIAI
FL 18MIAI	Porsche	Azul	387-6713-362	FL 176854
CA CATA17	Datsun	Blanco	391-3873-132	CA 13718
FL 171899	Ford	Rojo	391-3873-132	CA CATA17
NY BABYBLUE	Ferrari	Rojo	389-4816-381	NY BABYBLUE

(b) Caso de muestra después de una inserción

Figura 2.14. Casos del esquema de muestra.

bargo, es posible modificar ese estado. Por ejemplo, es posible que John Smith compre otro coche, un Ferrari rojo. Aquí añadimos un caso del registro COCHE para representar el coche nuevo y un caso del registro POSEE para representar la posesión de ese coche. El nuevo caso resultante del esquema de muestra aparece en la figura 2.14b.

Una forma de interpretar la relación entre esquema y caso es considerar el primero como una limitación para el segundo. De esta forma, entre todas las colecciones posibles de datos que pueden describir una realidad determinada sólo algunas de ellas son válidas con respecto al esquema; se dice que son **casos válidos** de ese esquema. Si se adopta este punto de vista, un objetivo importante del diseño conceptual de bases de datos es hacer una descripción semánticamente rica del esquema, que haga de filtro eficaz para casos no válidos de las bases de datos. Otra forma de ver las diferencias entre esquema y caso es considerar el esquema como un conocimiento *intensivo* y el caso como un conocimiento *extensivo*; el primero denota las propiedades estructurales de los datos; el segundo denota una asignación de valores a los datos.

La figura 2.15 representa las interrelaciones de modelo, esquema y caso. Se presentarán varios ejemplos de modelo, esquema y caso conforme se avance en el análisis de los modelos de datos.

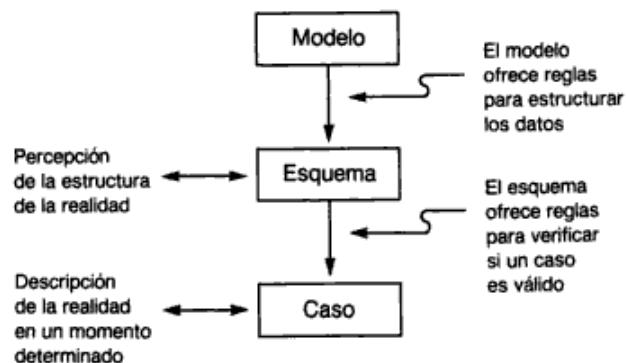


Figura 2.15. Las interrelaciones de modelo, esquema y caso.

2.3.1. Niveles múltiples de un sistema de bases de datos

El diseño de una base de datos no es el único campo para la aplicación de los modelos conceptuales. Un desarrollo significativo, a mediados de los años setenta, fue la propuesta de la Comisión SPARC del American National Standards Institute (Instituto Nacional Americano de Normas), popularmente conocida como propuesta ANSI/PARC. En la recomendación ANSI/SPARC cada sistema de bases de datos se organiza de acuerdo con tres niveles de descripción de datos: externo, conceptual e interno. El **nivel externo** describe los puntos de vista de grupos específicos de usuarios de la base de datos; presenta la información que es relevante para un grupo específico de usuarios. El **nivel conceptual**, también llamado *esquema de empresa*, ofrece una representación independiente de la máquina y de alto nivel de toda la base de datos. El **nivel interno** ofrece una descripción dependiente de la máquina de la implantación física de la base de datos. En esta arquitectura los modelos conceptuales son lenguajes para plasmar el nivel conceptual. Al ser ésta una arquitectura DBMS, los modelos conceptuales son «procesables», es decir, son comprendidos por el computador, el cual necesita varios traductores para establecer las correspondencias entre el nivel externo y los niveles conceptual e interno.

La importancia dada a esta aplicación de los modelos conceptuales ha disminuido con el tiempo. Si bien se han desarrollado prototipos importantes según las recomendaciones de ANSI/PARC, la tendencia actual de los DBMS comerciales ha sido, de alguna manera, contraria. Los nuevos DBMS comerciales no incluyen arquitecturas de niveles múltiples con complejas correspondencias entre los modelos; se inclinan por el uso de un modelo lógico simple, con lo que se alcanza mayor eficiencia. Sin embargo, cabe señalar que los DBMS comerciales ofrecen características de nivel externo o interno dentro de modelos lógicos.

En los años ochenta, los modelos conceptuales han encontrado otro campo importante de aplicación en los llamados **sistemas de diccionarios de datos**. Un diccionario de datos es un sistema de propósito especial, cuya función es definir

el contenido de la base de datos y de los programas de aplicación dentro de los sistemas de información. Puesto que la facilidad de lectura y la expresividad del modelo de datos son características importantes para el diccionario de datos, no sorprende el uso extenso de los modelos conceptuales en estos sistemas. Más adelante se hará hincapié en la importancia del uso de los modelos conceptuales para la documentación de las bases de datos.

2.3.2. Cualidades de los modelos conceptuales

Los modelos conceptuales deben ser buenas herramientas para representar la realidad; por esta razón, deben poseer las siguientes cualidades:

1. **Expresividad.** Los modelos conceptuales difieren en la elección y número de las distintas estructuras de modelado que ofrecen. En general, la disponibilidad de una amplia gama de conceptos hace posible una representación más extensa de la realidad; por este motivo, los modelos más ricos en conceptos son también muy expresivos. Por ejemplo, la mayoría de los modelos conceptuales de datos hacen uso frecuente de la abstracción de generalización, lo que permite la representación directa en el esquema de una gran variedad de restricciones de integridad, es decir, aserciones que permiten la selección de casos válidos del esquema de base de datos.
2. **Simplicidad.** Un modelo conceptual debe ser simple, para que un esquema creado con ese modelo sea fácil de entender por los diseñadores y usuarios de la aplicación de bases de datos. Obsérvese, sin embargo, que la simplicidad y la expresividad son objetivos en conflicto; si un modelo es semánticamente rico, es probable que no sea simple.
3. **Minimalidad.** Esta propiedad se consigue si cada concepto presente en el modelo tiene un significado distinto con respecto a todos los demás (en otras palabras, si ningún concepto puede expresarse mediante otros conceptos).
4. **Formalidad.** Los esquemas creados usando modelos conceptuales de datos representan una especificación formal de los datos. La formalidad requiere que todos los conceptos del modelo tengan una interpretación única, precisa y bien definida. Los conceptos formales pueden manipularse matemáticamente.

En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada; algunas de estas propiedades deben expresarse mediante aserciones que complementen el esquema. Sin embargo, el número de aserciones necesarias puede hacerse arbitrariamente pequeño, al incorporar conceptos más expresivos en el modelo. La selección del nivel apropiado de complejidad para un modelo es una decisión difícil: sería conveniente el uso de un modelo conceptual que incorporase varios ingredientes sin detrimento de su simplicidad y manejabilidad. El grado de minimalidad de un modelo implica también un equilibrio, porque la disponibilidad de un mayor y más rico conjunto de

conceptos, posiblemente superpuestos, ayuda al analista a percibir y modelar la realidad.

2.3.3. Propiedades de las representaciones gráficas

Los modelos de datos suelen describirse mediante representaciones lingüísticas y gráficas. En el apartado anterior se mostró una representación gráfica del modelo de muestra. El éxito de un modelo depende con frecuencia del éxito de su representación gráfica, que debe poseer las siguientes cualidades:

1. **Compleción gráfica.** Un modelo es gráficamente completo si todos sus conceptos poseen una representación gráfica; de otro modo, la representación gráfica tiene que complementarse con una representación lingüística.
2. **Facilidad de lectura.** Un modelo es fácil de leer si cada concepto se representa con un símbolo gráfico claramente distingible del resto de los símbolos gráficos.

Obsérvese que se tratan ahora las propiedades de los modelos y no de los esquemas; es responsabilidad del analista crear esquemas legibles. Se abordará este tema más adelante.

2.4. El modelo de entidades-interrelaciones

Este libro se concentra en el modelo de entidades-interrelaciones (ER), el modelo de datos más ampliamente usado para el diseño conceptual de bases de datos. El modelo fue introducido por Peter Chen en 1976, y se ha hecho cada vez más popular. Se han organizado varias conferencias sobre las aplicaciones del modelo ER en el diseño de bases de datos y en el de software en general. En 1988 el ANSI seleccionó el modelo ER como el modelo estándar para los sistemas de diccionarios de recursos de información (IRDS, Information Resource Dictionary Systems).

Originalmente, el modelo ER incluía sólo los conceptos de entidad, interrelación y atributos; más tarde, otros conceptos, como los atributos compuestos y las jerarquías de generalización, se agregaron como componentes del modelo ER mejorado. Se respetará este desarrollo cronológico y se presentarán elementos básicos y características avanzadas del modelo ER en dos apartados posteriores. Después de presentar el modelo ER, se mostrará cómo se representan las abstracciones en él y se ofrecerá una exposición crítica de sus cualidades.

2.4.1. Elementos básicos del modelo ER

Los conceptos básicos previstos por el modelo ER son entidades, interrelaciones y atributos. Nótese el uso de los términos *entidad* e *interrelación* para denotar

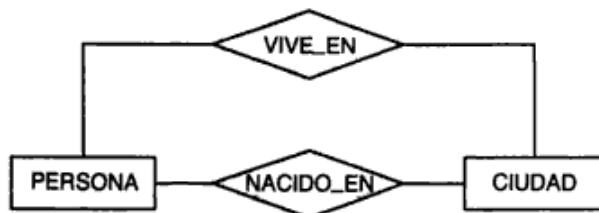


Figura 2.16. Parte de un esquema ER que representa las entidades PERSONA y CIUDAD y las interrelaciones NACIDO_EN y VIVE_EN.

clases de objetos; en la literatura, algunos autores usan los términos correspondientes *tipo de entidad* y *tipo de interrelación*.

Entidades. Las entidades representan clases de objetos de la realidad. PERSONA, HOMBRE, MUJER, EMPLEADO y CIUDAD son ejemplos de entidades para una base de datos de personal. Las entidades se representan gráficamente por medio de rectángulos, como muestra la figura 2.16.

Interrelaciones. Las interrelaciones representan agregaciones de dos o más entidades. Un ejemplo de interrelación binaria en la base de datos de personal es NACIDO_EN, que relaciona PERSONA y CIUDAD de nacimiento. Otra interrelación binaria entre las mismas entidades es VIVE_EN, que indica la ciudad donde la persona vive en la actualidad. Las interrelaciones se representan gráficamente con rombos, como se aprecia en la figura 2.16.

Las interrelaciones n-arias conectan más de dos entidades; por ejemplo, la interrelación SE_IMPARTE de la figura 2.17 es una interrelación ternaria que une las entidades CURSO, DIA y AULA, según se explicó en el apartado 2.2.2.

Los anillos son interrelaciones binarias que conectan una entidad consigo misma. Se conocen también como **interrelaciones recursivas**. Por ejemplo, la interrelación DIRIGE de la figura 2.18 conecta los directores con sus subordinados, ambos representados por la entidad EMPLEADO. Para distinguir entre los dos *papeles* de la entidad en la interrelación, se asocian dos *rótulos* con la entidad; en la figura 2.18 los dos rótulos son DIRECTOR_DE y SUBORDINADO_A.



Figura 2.17. Interrelación n-aria SE_IMPARTE.



Figura 2.18. La interrelación de anillo DIRIGE.

Cada interrelación tiene un significado específico; de ahí la necesidad de seleccionar nombres significativos para las interrelaciones. Por ejemplo, si el nombre EN fuese usado para nombrar una interrelación que conecta las entidades PERSONA y CIUDAD, el esquema no expresaría si la interrelación se refiere al lugar de nacimiento o la ciudad de residencia.

Considérese de nuevo el esquema en la figura 2.16. Un caso del esquema es como sigue:

$\text{PERSONA} = \{p_1, p_2, p_3\}$
 $\text{CIUDAD} = \{c_1, c_2, c_3\}$
 $\text{VIVE_EN} = \{\langle p_1, c_1 \rangle, \langle p_2, c_3 \rangle, \langle p_3, c_3 \rangle\}$
 $\text{NACIDO_EN} = \{\langle p_1, c_1 \rangle, \langle p_3, c_1 \rangle\}$

La entrada para p_2 en NACIDO_EN puede faltar si la persona p_2 nació en una ciudad diferente de c_1 , c_2 y c_3 . Adviértase que después de la introducción de las entidades e interrelaciones sólo se puede indicar casos de ellas; no se puede describir las propiedades de cada caso. Esto será posible después de introducir los atributos.

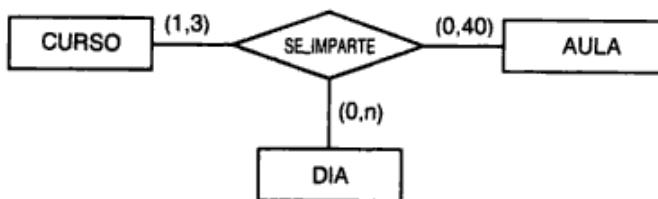
Las interrelaciones se caracterizan en términos de cardinalidad mínima y máxima, como se trató en el apartado 2.2. En el ejemplo anterior,

$\text{card-mín}(\text{PERSONA}, \text{VIVE_EN}) = 1$,
 $\text{card-máx}(\text{PERSONA}, \text{VIVE_EN}) = 1$,
 $\text{card-mín}(\text{CIUDAD}, \text{VIVE_EN}) = 0$, y
 $\text{card-máx}(\text{CIUDAD}, \text{VIVE_EN}) = n$.

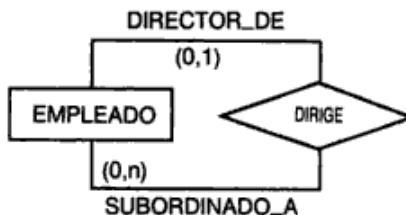
Con base en las cardinalidades antes expuestas, VIVE_EN es una interrelación de uno a muchos entre PERSONA y CIUDAD. La participación de PERSONA en la interrelación es obligatoria, mientras que la participación de ciudad en la interrelación es opcional. Se resume la cardinalidad mínima y máxima en un par de valores: $\text{card}(\text{PERSONA}, \text{VIVE_EN}) = (1,1)$ y $\text{card}(\text{CIUDAD}, \text{VIVE_EN}) = (0,n)$. Cada par está representado en el esquema (Fig. 2.19a), cerca de la conexión entre la entidad (rectángulo) y la interrelación (rombo). Otras representaciones gráficas en la literatura usan líneas dobles que inciden en el rombo para representar $\text{card-máx} = n$ y una línea sencilla que incide en el rombo para represen-



(a) La interrelación VIVE_EN



(b) La interrelación SE_IMPARTE



(c) La interrelación DIRIGE



(d) La interrelación EMBARQUE

Figura 2.19. Ejemplos de interrelaciones en el modelo ER.

tar card-máx = 1; semejantes representaciones gráficas no incluyen cardinalidades mínimas. La figura 2.19 describe algunas interrelaciones y señala las correspondientes cardinalidades mínimas y máximas para:

1. La interrelación VIVE_EN (ya mencionada).
2. La interrelación SE_IMPARTE (tratada en el apartado 2.2.2).
3. La interrelación DIRIGE (una interrelación de uno a muchos, puesto que cada director dirige a varios empleados y cada empleado tiene sólo un director, la participación en la interrelación es opcional).
4. La interrelación EMBARQUE entre un PEDIDO y la correspondiente TARJETA_DE_EMBARQUE (una interrelación de uno a uno, puesto que cada pedido

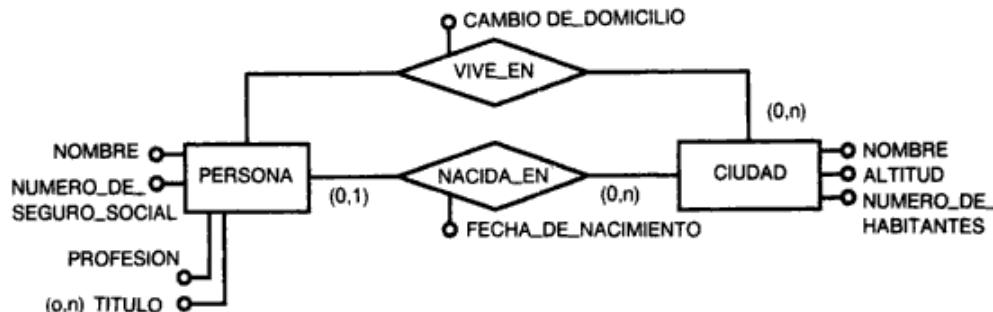


Figura 2.20. Un esquema ER con entidades, interrelaciones y atributos.

está relacionado opcionalmente con una tarjeta y cada tarjeta está obligatoriamente relacionada con un pedido).

Atributos. Los atributos representan las propiedades básicas de las entidades o interrelaciones. Toda la información extensiva es portada por los atributos.

Se puede añadir atributos al esquema de la figura 2.16. Los atributos de PERSONA son: NOMBRE, NUMERO_DE_SEGURO_SOCIAL, PROFESION, TITULO. Los atributos de CIUDAD son: NOMBRE, ALTITUD, NUMERO_DE_HABITANTES. El único atributo de VIVE_EN es FECHA_DE_CAMBIO_DE_DOMICILIO, con la fecha en que la persona se mudó a la ciudad. El único atributo de NACIDO_EN es FECHA_DE_NACIMIENTO. El esquema ER resultante se muestra en la figura 2.20.

Como las interrelaciones, los atributos se caracterizan por su cardinalidad mínima y máxima. La cardinalidad mínima indica el número mínimo de valores de atributos asociados con cada caso de entidad o interrelación. Sea A un atributo de la entidad E; si $\text{card-mín}(A,E) = 0$, el atributo es opcional y puede estar no especificado (nulo) en algunos casos de la entidad. Si, por el contrario, $\text{card-mín}(A,E) = 1$, el atributo es obligatorio y al menos un valor del atributo debe especificarse para todos los casos de la entidad. Las mismas definiciones se aplican a atributos de interrelaciones. En el ejemplo, NOMBRE, NUMERO_DE_SEGURO_SOCIAL y PROFESION son atributos obligatorios; por ende, no se aceptará la inclusión de una persona en la base de datos cuyo NOMBRE, NUMERO_DE_SEGURO_SOCIAL y PROFESION no se especifiquen. TITULO, en cambio, es opcional y se aceptará la inserción de una persona cuyo título no se especifique (sea nulo).

La cardinalidad máxima indica el número máximo de valores de atributos asociados con cada entidad o interrelación. Sea A un atributo de la entidad E; si $\text{card-máx}(A,E)=1$, el atributo es **monovalente**; si $\text{card-máx}(A,E) > 1$, el atributo es **polivalente**. Las mismas definiciones se aplican a los atributos de las interrelaciones. En el ejemplo, NOMBRE, NUMERO_DE_SEGURO_SOCIAL y PROFESION son **monovalentes**, pues cada PERSONA tiene un NOMBRE, un NUMERO_DE_SEGURO_SOCIAL y una PROFESION. El atributo TITULO es **polivalente**.

lente, porque cada persona puede tener múltiples grados: ds (diploma de secundaria), bh (bachiller en humanidades), mc (maestro en ciencias) en física, d (doctor), en informática, etcétera.

La cardinalidad de los atributos es el par (card-mín, card-máx); como en el caso de las interrelaciones, se representa en el esquema junto al atributo. El valor que se da con más frecuencia es (1,1), que se toma como valor por omisión y, por tanto, se omite en las figuras.

Cada atributo se asocia a un **dominio** particular, es decir, el conjunto de valores legítimos para ese atributo. Las declaraciones de dominio se asemejan a las declaraciones de tipo en los lenguajes convencionales de programación. Un atributo **simple** se define sobre un dominio.

Un ejemplo de caso del esquema de base de datos de la figura 2.20 es el siguiente:

```

PERSONA = {p1: <JOHN, 345-8798-564, ESTUDIANTE, ()>,
            p2: <SUE, 675-6756-343, GERENTE, (MC, Ing, D)>,
            p3: <MARTIN, 676-453-8482, GRANJERO, (DS)>}

CIUDAD = {c1: <ROMA, 100, 3000000>,
           c2: <NUEVA-YORK, 0, 9000000>,
           c3: <ATLANTA, 100, 2000000>}

VIVE_EN = {<p1,c1: <1-02-80> >,
            <p2,c3: <7-23-83> >,
            <p3,c3: <6-04-81> > }

NACIDO_EN = {p1,c1: <1-05-55> >,
              <p3,c1: <6-14-35> }

```

Obsérvese que los valores de los atributos polivalentes están encerrados entre paréntesis; el estudiante John no está asociado con ningún título.

El esquema de la figura 2.20 modela una realidad que incluye personas, ciudades, nacimientos y residencias. En particular, el diseñador ha percibido las ciudades como datos primarios de interés y ha decidido modelarlos por medio de una entidad. Supóngase ahora una nueva situación, en la que no interesan las características de las ciudades, como el número de habitantes o la altitud, pero sí se consideran, en cambio, la ciudad de nacimiento y de residencia como dos propiedades elementales de la persona. Entonces, todo el esquema de la figura 2.20 se reducirá a una sola entidad, PERSONA, con los atributos NOMBRE, NUMERO_DE_SEGURO_SOCIAL, PROFESION, TITULO, CIUDAD_DE_NACIMIENTO, FECHA_DE_NACIMIENTO, CIUDAD_DE_RESIDENCIA y FECHA_DE_CAMBIO_DE_DOMICILIO. Este ejemplo muestra que la decisión de usar entidades, interrelaciones o atributos para modelar algunos aspectos de la realidad es bastante delicada: existen muchos esquemas similares que modelan diferentes vistas de la misma realidad. En este libro no sólo se presentan los modelos de datos, sino también metodologías que ayudan al lector a seleccionar el esquema más conveniente para describir la realidad, de entre todas las representaciones alternativas.

```

Esquema: PERSONAL

Entidad: PERSONA
Atributos: NOMBRE: texto(50)
           NUMERO_DE_SEGURO_SOCIAL: texto(12)
           PROFESION: texto(20)
           (0,n)TITULO: texto(20)

Entidad: CIUDAD
Atributos: NOMBRE: texto(30)
           ALTITUD: entero
           NUMERO_DE_HABITANTES: entero

Interrelación: VIVE_EN
Entidades conectadas: (0,n)CIUDAD
                      (1,1)PERSONA
Atributos: FECHA_DE_CAMBIO_DE_DOMICILIO: fecha

Interrelación: NACIDO_EN
Entidades conectadas: (0,n)CIUDAD
                      (0,1)PERSONA
Atributos: FECHA_DE_NACIMIENTO: fecha

```

Figura 2.21. Definición lingüística de un esquema conceptual.

Se concluye este apartado mostrando en la figura 2.21 una representación lingüística para todos los conceptos presentados. La notación se explica por sí misma; se presentará una gramática BNF (forma de Backus-Naur) para este lenguaje de definición de esquemas en el próximo apartado.

2.4.2. Otros elementos del modelo ER

Entre los demás elementos del modelo ER están las jerarquías de generalización, los subconjuntos, los atributos compuestos y los identificadores.

Jerarquías de generalización. En el modelo ER es posible establecer jerarquías de generalización entre las entidades. Una entidad E es una **generalización** de un grupo de entidades E_1, E_2, \dots, E_n si cada objeto de las clases E_1, E_2, \dots, E_n es también un objeto de la clase E. Una generalización en el modelo ER expresa la abstracción de generalización expuesta en el apartado 2.1.3. La representación esquemática de las generalizaciones se muestra en la figura 2.22. La flecha señala la entidad generalizada.

Cada entidad puede participar en múltiples generalizaciones, posiblemente en el papel de entidad genérica con respecto a una generalización y en el papel de entidad subconjunto con respecto a otra generalización. La figura 2.23 presenta una jerarquía de generalización compleja para la entidad PERSONA. Lo opuesto a la generalización se denomina **especialización**.

Las jerarquías de generalización se caracterizan por la propiedad de cober-

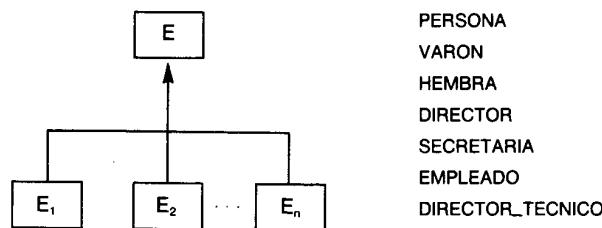


Figura 2.22. Representación de una generalización en el modelo ER.

tura, la cual se explicó en el apartado 2.2.3. Recuérdese que cada generalización puede ser total (t) o parcial (p) y exclusiva (e) o superpuesta (s). El par que se da con más frecuencia es (t,e), que se considera como el valor por omisión y, por tanto, se omite en las figuras.

En el ejemplo de la figura 2.23, la cobertura de la generalización es como sigue:

1. La generalización basada en el sexo es total y exclusiva. Con frecuencia se nombra la generalización en términos de lo que le sirve como base de definición. Según esto, tal generalización puede llamarse SEXO, y se usará este nombre en la descripción lingüística (Fig. 2.30).
2. Si se supone que el dominio de aplicación incluye personas que no son empleados ni secretarias ni directores, la generalización basada en el papel desempeñado es parcial y exclusiva.
3. Si se supone que los empleados pueden tener más de un tipo de trabajo y que algunos tienen un tipo de trabajo distinto de los que se representan ex-

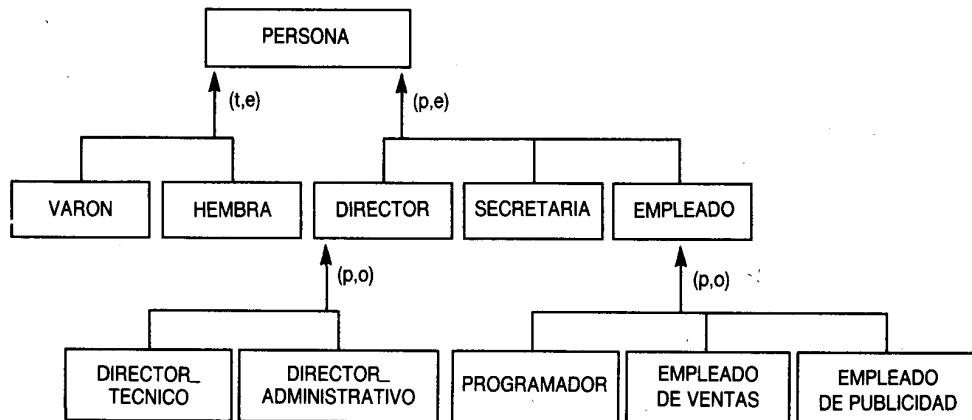


Figura 2.23. Jerarquía de generalización para la entidad PERSONA.

plícitamente, la generalización que se basa en el tipo de trabajo es parcial y superpuesta.

4. Si se supone que todos los directores tienen un papel directivo pero también que algunos pueden tener papeles tanto técnicos como administrativos, la generalización basada en el papel administrativo es total y superpuesta.

Recuérdese la propiedad fundamental de la abstracción de generalización: todas las propiedades de la entidad genérica son heredadas por las entidades subconjunto. En términos del modelo ER, esto significa que cada atributo, interrelación o generalización definido para la entidad genérica será heredado automáticamente por todas las entidades subconjunto de la generalización. Esta propiedad es importante, porque permite construir jerarquías de generalización estructuradas.

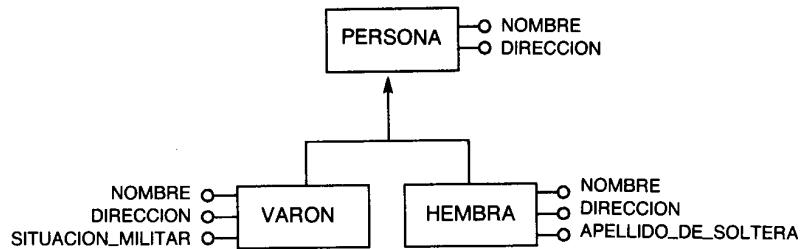
Considérese la figura 2.24a. La propiedad de herencia establece que los atributos NOMBRE y DIRECCION de PERSONA son también atributos de VARON y HEMBRA; luego, pueden ser eliminados de las entidades subconjunto, simplificando el esquema, como muestra la figura 2.24c.

Por otro lado, considérese el esquema de la figura 2.24b: éste representa una colocación impropia de los atributos dentro de la generalización. Ciertamente, el atributo común NOMBRE, que es una propiedad general de PERSONA, debe situarse más arriba en la jerarquía. En cambio, es obvio que SITUACION_MILITAR y APELLIDO y APELLIDO_DE_SOLTERA atan a VARON y HEMBRA, respectivamente, y deben situarse más abajo, como muestra el esquema de la figura 2.24c. Hay que hacer notar que la cardinalidad mínima de los dos atributos era 0 por la colocación incorrecta de los atributos en la generalización; se convierte en 1 después de la modificación.

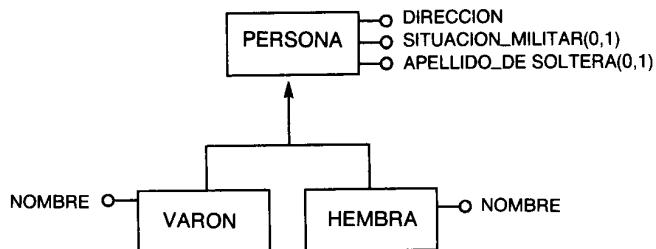
Subconjuntos. Un subconjunto es un caso particular de jerarquía de generalización, con una sola entidad subconjunto. Trataremos por separado los subconjuntos porque la cobertura de un subconjunto es claramente parcial y exclusiva y no necesita definirse. Se representan los subconjuntos con una flecha que une la entidad genérica a la entidad subconjunto y apunta hacia la entidad genérica, como en la figura 2.25. Una entidad subconjunto puede tener atributos adicionales, como FECHA_DE_CONFIRMACION para TRABAJADOR_FIJO.

Atributos compuestos. Los atributos compuestos son grupos de atributos que tienen afinidad en cuanto a su significado o a su uso. Por ejemplo, el atributo compuesto DIRECCION abarca el grupo de atributos CALLE, CIUDAD, PROVINCIA, CODIGO_POSTAL y PAIS. Los atributos compuestos se representan con óvalos, como muestra la figura 2.26.

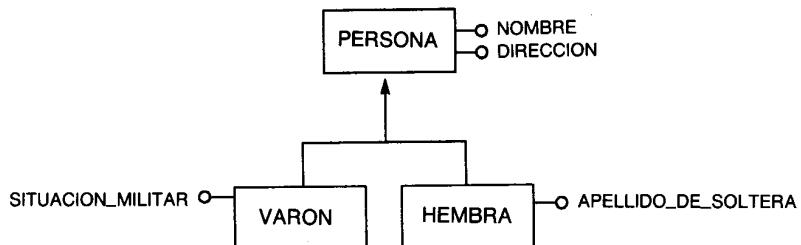
Las cardinalidades mínima y máxima se aplican a los atributos compuestos con las mismas definiciones dadas para los atributos simples. Nótese, empero, que asignar cardinalidades mínimas y máximas a los atributos compuestos agrega más capacidades de modelado en comparación con la asignación de cardinalidades a cada atributo individual. En el ejemplo de la figura 2.26, se afirma que una misma persona puede tener varias direcciones, y que cada dirección está



(a) Representación incorrecta



(b) Representación incorrecta



(c) Representación correcta

Figura 2.24. Transformaciones de las jerarquías de generalización debido a la propiedad de herencia.

compuesta por una calle, una ciudad, una provincia, un país y, opcionalmente, un código postal. En cambio, si se usan cinco atributos independientes, sólo se podrá afirmar que cada uno de tales atributos es polivalente de manera independiente, y se tendrá una menor capacidad expresiva.

Identificadores. Un identificador de una entidad E es un grupo de atributos o de entidades relacionados con E, que tienen la propiedad de determinar

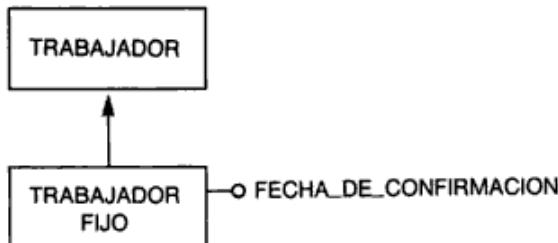


Figura 2.25. Ejemplo de subconjunto.

en forma única todos los casos de E. Desde un punto de vista terminológico, los identificadores se llaman a veces en la literatura *claves* o *claves candidatas*.

De un modo más formal, sea E una entidad, sean A_1, \dots, A_n atributos monovalentes obligatorios de E, sean E_1, \dots, E_m otras entidades vinculadas a E por interrelaciones binarias obligatorias, de uno a uno o de muchos a uno, R_1, \dots, R_m , (es decir, tales que $\text{card-mín}(E, R_i) = \text{card-máx}(E, R_i) = 1$). Considérese como un posible identificador el conjunto $I = \{A_1, \dots, A_n, E_1, \dots, E_m, n \geq 0, m \geq 0, n + m \geq 1\}$. El **valor** del identificador para un caso particular de la entidad e se define como el conjunto de todos los valores de los atributos A_i , $i = 1, 2, \dots, n$, y todos los casos de las entidades E_j , $j = 1, 2, \dots, m$, vinculadas a e, con $i \leq n$, $j \leq m$. A causa de la suposición de considerar atributos monovalentes obligatorios o interrelaciones obligatorias con una cardinalidad máxima fija en 1, cada caso de E corresponde a un valor del atributo A_i o bien a un caso de la entidad E_j , para $i \leq n$, $j \leq m$.

I es un identificador de E si se cumplen las siguientes condiciones:

1. No pueden existir dos casos de E con el mismo valor del identificador.
2. Si se omite cualquier atributo A_i o entidad E_j del identificador, la condición 1 deja de cumplirse.

Nótese que por causa de nuestras suposiciones acerca de la cardinalidad de los atributos o entidades dentro de las interrelaciones que forman un identificador, el «valor» de un identificador está siempre bien definido. En otras palabras, por cada caso de la entidad E, existe un y sólo un valor del atributo A_i , y un y sólo un caso de la entidad E_j , relacionados con ese caso de E; los atributos que pueden tener valores nulos no pueden participar en un identificador.



Figura 2.26. Ejemplo de atributo compuesto y polivalente.

Cada entidad puede tener múltiples identificadores alternativos. Se clasifican los identificadores como sigue:

1. Un identificador es **simple** si $n + m = 1$; es **compuesto** si $n + m > 1$.
2. Un identificador es **interno** si $m = 0$; es **externo** si $n = 0$.
3. Un identificador es **mixto** si $n > 0$ y $m > 0$.

Por lo general, se prefiere los identificadores internos a los externos porque son más simples de entender y usar. Por la misma razón se prefiere los identificadores simples a los compuestos. Al final del proceso de diseño, se requiere que cada entidad sea provista de al menos un identificador.

Es importante evitar la circularidad en la definición de los identificadores (por ejemplo, el uso de la entidad E_i en la identificación de la entidad E_j y de la entidad E_j en la identificación de la entidad E_i). A fin de evitar la circularidad, siempre que una entidad E_j esté implicada en la identificación de la entidad E , E_j deberá poseer un identificador adecuado que no dependa de E . Esto se consigue en la práctica comenzando el proceso de identificación con las entidades que puedan identificarse internamente (estas entidades se llaman a veces **entidades fuertes**) y creando después identificadores para las entidades que sólo posean identificadores externos (a veces llamadas **entidades débiles**).

La figura 2.27 presenta varios ejemplos de identificadores. En todos los casos, el símbolo gráfico para la identificación es un círculo negro; sin embargo, cada tipo de identificador requiere una representación gráfica diferente. En la figura 2.27a, **NUMERO_DE_SEGURO_SOCIAL** es un identificador simple e interno de **PERSONA**, lo que se representa coloreando en negro el símbolo del atributo correspondiente.

En la figura 2.27b, **NOMBRE**, **FECHA_DE_NACIMIENTO**, **NOMBRE_DEL_PADRE** y **CIUDAD_DE_RESIDENCIA** forman un identificador compuesto e interno de **PERSONA**. Los identificadores compuestos están representados gráficamente por un segmento de línea que une los dos o más elementos que confieren identificación. Un círculo negro señala la intersección entre el segmento y cada elemento del identificador. Si al identificador se le asigna un nombre, uno de los extremos del segmento tiene un círculo negro y un nombre. En este ejemplo, el nombre es **IDENTIFICADOR_DE_PERSONA**.

Considérese la entidad **EMPLEADO** (Fig. 2.27c), conectada a la entidad departamento por la interrelación **TRABAJA_EN**, con card-máx (**EMPLEADO**, **TRABAJA_EN**) = 1. Entonces, la entidad **DEPARTAMENTO** y el atributo **NUMERO_DE_EMPLEADO_POR_DEPARTAMENTO** constituyen un identificador externo, compuesto y mixto de **EMPLEADO**.

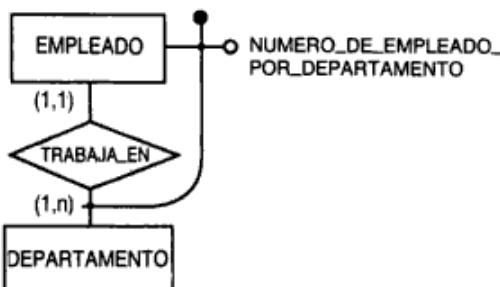
Ahora considérese la entidad **DETALLE_PEDIDO** (Fig. 2.27d). Supóngase que cada **DETALLE_PEDIDO** está conectado a la entidad **PRODUCTO** mediante la interrelación **PARA** y a la entidad **CABECERA_PEDIDO** mediante la interrelación **DE**, con card (**DETALLE_PEDIDO**, **DE**) = card (**DETALLE_PEDIDO**, **PARA**) = (1,1). Se supone también que dos **DETALLE_PEDIDO** no pueden relacionarse con el mismo producto dentro del mismo pedido. Entonces, el par de entidades **CABECERA-**



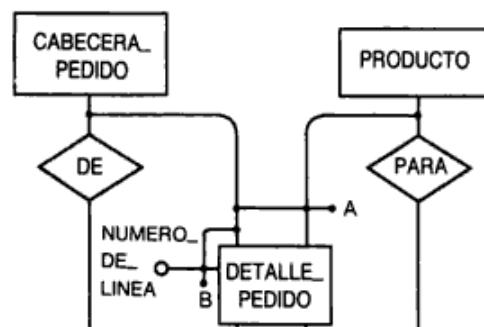
(a) Identificador simple e interno



(b) Identificador compuesto e interno



(c) Identificador compuesto, externo y mixto



(d) Identificadores de la entidad DETALLE_PEDIDO

Figura 2.27. Identificadores en el modelo ER.

_PEDIDO (mediante DE) y producto (mediante PARA) componen un identificador externo y compuesto, llamado A, de DETALLE_PEDIDO. Como alternativa para la entidad DETALLE_PEDIDO, supóngase que cada DETALLE_PEDIDO se numera progresivamente dentro de cada pedido por el atributo NUMERO_DE_LINEA; entonces, el par CABECERA_PEDIDO (mediante DE) y NUMERO_DE_LINEA es un identificador externo, mixto y compuesto, llamado B, de DETALLE_PEDIDO.

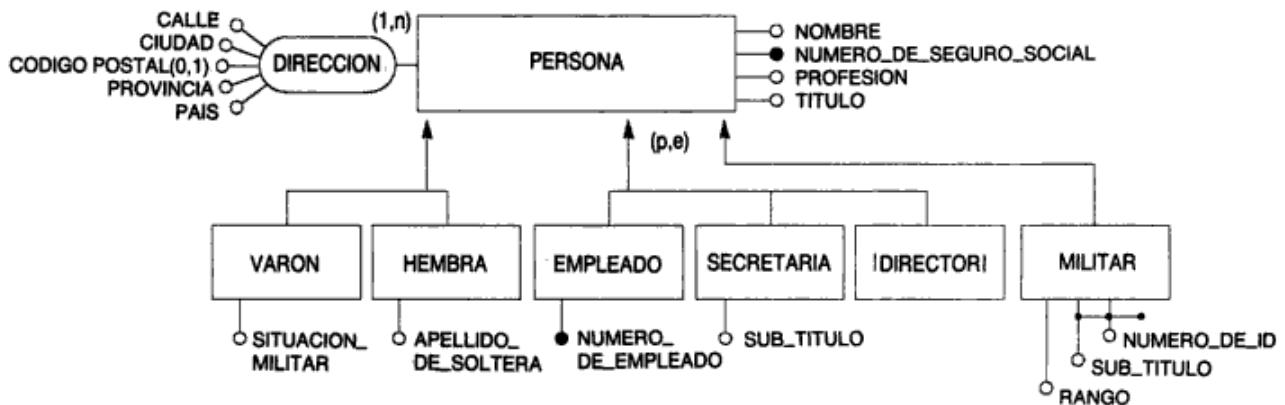


Figura 2.28. Jerarquías de generalización, atributos compuestos e identificadores en el modelo ER.

Puesto que la identificación es una propiedad de las entidades, es una de las propiedades heredadas en las generalizaciones o subconjuntos: el identificador de la entidad genérica es, asimismo, un identificador de las entidades subconjunto. Las entidades subconjunto pueden tener identificadores adicionales. En la figura 2.28, cada persona se identifica por el número de seguro social; cada empleado se identifica por un número de empleado, asignado dentro de la compañía; cada militar se identifica por el nombre de la división y el número de identificación dentro de la división.

Concluimos este apartado mostrando una representación lingüística del modelo ER. La figura 2.29 muestra la gramática BNF³ completa para el lenguaje de definición de esquemas. Se usan estas convenciones: los corchetes denotan opción, las claves indican repetición, el prefijo *lista_de* indica una secuencia de elementos separados por comas y el prefijo *nombre_de* denota los identificadores. Las reglas para los no terminales **ENTERO** y **VALOR** se omiten. Cada definición de esquema se divide en una secuencia de definiciones de entidades, definiciones de jerarquías de generalización y definiciones de interrelaciones. Cada definición se explica por sí misma. Obsérvese que una generalización se define en términos de padre (entidad) e hijos (entidades); el padre representa la entidad superconjunto.

La figura 2.30 presenta una definición lingüística, de acuerdo con la gramática BNF del lenguaje CSDL dada en la figura 2.29, del esquema de la figura 2.28. La figura 2.21 contiene otro ejemplo de definición lingüística de un esquema, de acuerdo con la gramática BNF de la figura 2.29. Para finalizar, la figura 2.31 resume los símbolos gráficos usados en el modelo ER.

⁴ En la convención BNF, una gramática se describe con un conjunto de reglas o «producciones». Los términos que aparecen en el lado izquierdo de al menos una regla se llaman *no terminales*; los que no aparecen en el lado izquierdo de ninguna regla son *terminales*. Las barras verticales en el lado derecho de una regla muestran formas alternativas de satisfacer el no-terminal del lado izquierdo. Si se desea mayores detalles acerca de cómo leer una gramática BNF, consultese un texto sobre lenguajes de programación o compiladores.

```

ESQUEMA→Esquema: NOMBRE_DE_ESQUEMA
SECCION_DE_ENTIDADES
SECCION_DE_GENERALIZACIONES
SECCION_DE_INTERRELACIONES

SECCION_DE_ENTIDADES→{DECL_DE_ENTIDAD}

DECL_DE_ENTIDAD→Entidad: NOMBRE_DE_ENTIDAD
[SECCION_DE_ATRIBUTOS]
[SECCION_DE_ATRIBUTOS_COMPUUESTOS]
[SECCION_DE_IDENTIFICADORES]

SECCION_DE_ATRIBUTOS→Atributos: {DECL_DE_ATRIBUTO}
DECL_DE_ATRIBUTOS→[(CARD_MIN,CARD_MAX)]NOMBRE_DE_ATRIBUTO[:DECL_DES_TIPO]
CARD_MIN→0 | 1 | ENTERO
CARD_MAX→1 | n | ENTERO
DECL_DES_TIPO→entero | real | booleano | texto(ENTERO) | enumeración (LISTA_DE_VALORES)

SECCION_DE_ATRIBUTOS_COMPUUESTOS: Atributos compuestos:{DECL_DE_ATRIBUTO}
DECL_ATR_COMP→[(CARD_MIN,CARD_MAX)]NOMBRE_DE_ATR_COMP de
{DECL_DE_ATRIBUTO}

SECCION_DE_IDENTIFICADORES→Identificadores:{DECL_DE_IDENTIFICADOR}
DECL_DE_IDENTIFICADOR→LISTA_DE_IDENTIFICADORES
LISTA_DE_IDENTIFICADORES→{IDENTIFICADOR}
IDENTIFICADOR→NOMBRE_DE_ATRIBUTO | NOMBRE_DE_ENTIDAD (mediante
NOMBRE_DE_INTERRELACION)

SECCION_DE_GENERALIZACIONES→ [SECCION_DE_JERARQ_DE_GEN]
[SECCION_DE_SUBCONJUNTOS]

SECCION_DE_JERARQ_DE_GEN→(DECL_DE_JERARQ_DE_GEN)
DECL_DE_JERARQ_DE_GEN→Generalización([COBERTURA1,COBERTURA2]);
NOMBRE_DE_GENERALIZACION
Padre: NOMBRE_DE_ENTIDAD
Hijos: LISTA_DE_NOMBRE_DE_ENTIDAD

COBERTURA1→p | t
COBERTURA→e | o

SECCION_DE_SUBCONJUNTOS→{DECL_DE_SUBCONJUNTO}
DECL_DE_SUBCONJUNTO→Subconjunto: NOMBRE_DE_ENTIDAD de NOMBRE_DE_ENTIDAD

SECCION_DE_INTERRELACIONES→{DECL_DE_INTERRELACION}
DECL_DE_INTERRELACION→interrelación: NOMBRE_DE_INTERRELACION
Entidades conectadas: {DECL_DE_ENT_CONEC}
Atributos: {DECL_DE_ATRIBUTO}

DECL_DE_ENT_CONEC→[(CARD_MIN,CARD_MAX)]NOMBRE_DE_ENTIDAD

```

Figura 2.29. Gramática BNF del lenguaje de definición de esquemas conceptuales (CSDL, *Conceptual Schema Definition Language*).

2.4.3. Mecanismos de abstracción en el modelo ER

Es de interés mostrar cómo las tres abstracciones presentadas en el apartado 2.1 son asimiladas por los conceptos del modelo ER.

```

Esquema: PERSONAL
Entidad: PERSONA
    Atributos: NOMBRE:texto(20)
                NUMERO_DE_SEGURO_SOCIAL:texto(12)
                PROFESION:texto(20)
                (1,n)TITULO:texto(20)
    Atributos compuestos: (0,n)DIRECCION de
                            CALLE:texto(30)
                            CIUDAD:texto(20)
                            (0,1)CODIGO_POSTAL:texto(5)
                            PROVINCIA:texto(2)
                            PAIS:texto(20)
    Identificadores: NUMERO_DE_SEGURO_SOCIAL

Entidad: VARON
    Atributos: SITUACION_MILITAR

Entidad: HEMBRA
    Atributos: NOMBRE_DE_SOLTERA:entero

Entidad: EMPLEADO
    Atributos: NUMERO_DE_EMPLEADO:entero
    Identificadores: NUMERO_DE_EMPLEADO

Entidad: SECRETARIA
    Atributos: SUB_TITULO:enumeración[MECANOGRAFA,ARCHIVISTAM]

Entidad: DIRECTOR

Entidad: MILITAR
    Atributos: RANGO
                DIVISION:texto(10)
                NUMERO_DE_ID: entero
    Identificadores: DIVISION,NUMERO_DE_ID

Generalización: SEXO
    Padre: PERSONA
    Hijos: VARON,HEMBRA

Generalización (p,e): TIPO_DE_TRABAJO
    Padre: PERSONA
    Hijos: EMPLEADO,SECRETARIA,DIRECTOR

Subconjunto: MILITAR de PERSONA

```

Figura 2.30. Ejemplo de la definición del esquema ER de la figura 2.28, usando el CSDL de la figura 2.29.

Abstracción de clasificación. Los tres conceptos básicos del modelo ER se desarrollan como aplicaciones de la abstracción de clasificación:

1. Una *entidad* es una clase de objetos del mundo real con propiedades comunes.
2. Una interrelación es una clase de hechos atómicos (elementales) que relacionan dos o más entidades.

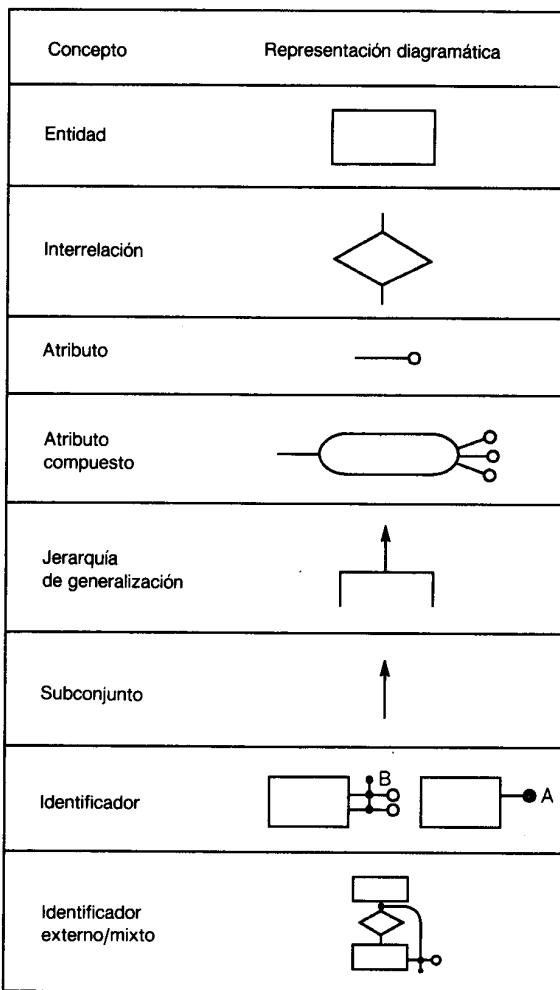


Figura 2.31. Símbolos gráficos usados en el modelo ER.

3. Un *atributo* es una clase de valores que representan propiedades atómicas de las entidades o interrelaciones.

Abstracción de agregación. Tres tipos de agregaciones caracterizan el modelo ER:

1. Una *entidad* es una agregación de atributos.
2. Una *interrelación* es una agregación de entidades y atributos.
3. Un *atributo compuesto* es una agregación de atributos.

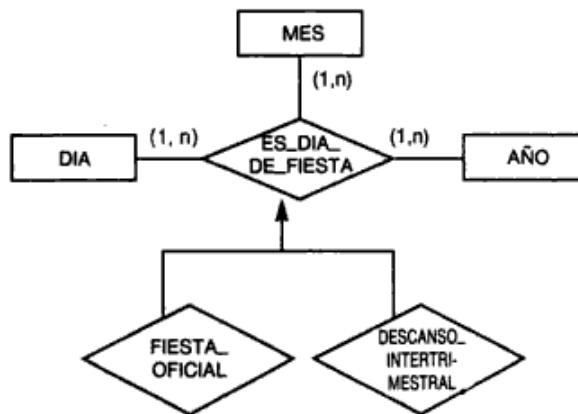


Figura 2.32. Abstracción de generalización aplicada a interrelaciones.

Abstracción de generalización. La abstracción de generalización la encarnan las jerarquías de generalización y los subconjuntos. Usualmente, sólo se aplica a *entidades*, aunque algunas ampliaciones del modelo ER aplican la abstracción de generalización también a interrelaciones o atributos. Un ejemplo de una ampliación así se muestra en la figura 2.32, donde la interrelación *ES_DIA_DE_FIESTA* es una generalización de *DESCANSO_INTERTRIMESTRAL* y *FIESTA_OFICIAL*. A partir de ahora no se incluirá esta posibilidad en nuestro modelo, por su uso poco frecuente.

2.4.4. Cualidades del modelo de entidades-interrelaciones

El modelo ER tiene muchos seguidores, pero también muchos críticos. Los seguidores aprecian su riqueza de conceptos, lo que le convierte en un modelo realmente potente para la descripción de la realidad; los críticos rechazan el modelo ER precisamente por esta riqueza de conceptos que pone en peligro su simplicidad y minimalidad. Se puede tratar de evaluar el modelo ER exponiendo sus cualidades, ya definidas en los apartados 2.4.2 y 2.4.3.

La expresividad del modelo ER es muy buena. El apartado anterior ha mostrado que el modelo ER incorpora los tres mecanismos de abstracción de varias formas.

Es cierto que el modelo ER no es muy sencillo. En particular, las propiedades de cardinalidad e identificación son difíciles de entender y usar. Sin embargo, estos rasgos son muy útiles para entender las propiedades estructurales de los esquemas de bases de datos. Por ello, el esfuerzo requerido del lector será a la larga muy bien recompensado.

La inclusión de interrelaciones n-arias en el modelo ER es criticada a menudo por los defensores de los llamados modelos binarios, quienes opinan que las interrelaciones debieran ser sólo binarias. Es cierto que muchos grandes es-

quemas de bases de datos, con cientos de entidades e interrelaciones, no incluyen ninguna interrelación n-aria. A pesar de esto, las interrelaciones n-arias son útiles en algunas situaciones que son de hecho modeladas «con naturalidad» por ellas; la descomposición de una interrelación n-aria en múltiples interrelaciones binarias anula el propósito original.

A pesar de las apariencias, el modelo ER es mínimo. Ningún concepto puede sustituirse por otra combinación de conceptos, con la única excepción de los atributos compuestos. De hecho, las entidades, interrelaciones, atributos y jerarquías de generalización son ciertamente mínimos. Los subconjuntos son sólo un caso especial de jerarquías de generalización; así que no invalidan esta afirmación. Las restricciones de cardinalidad y las propiedades de cobertura son también mínimas, y las propiedades de identificación no se pueden deducir de las propiedades de cardinalidad, por lo que la identificación es también mínima. Los atributos compuestos pueden modelarse mediante el uso adecuado de entidades e interrelaciones; sin embargo, en la práctica, los atributos compuestos son muy útiles para desglosar entidades complejas, con un gran número de atributos (las entidades de la realidad pueden contener cientos de atributos).

El lector no debe confundirse por el hecho de que una misma realidad pueda describirse de varias formas, esto no afecta la minimalidad del modelo. Se ha observado que la misma información puede representarse como una entidad o como una interrelación; de manera similar, algunas veces la misma información puede modelarse como entidad o como atributo. No obstante, los mismos problemas surgen con cualquier modelo de datos; se deben a la dificultad de coincidir en la forma de percibir la realidad. En la mayoría de los casos, dos diseñadores de bases de datos pueden percibir la realidad de dos formas distintas. Los esquemas finales difieren cuando las percepciones de los diseñadores difieren, sin importar el modelo usado para su creación.

El modelo ER está definido formalmente, como se ha mostrado en este apartado. También es gráficamente completo: cada uno de los conceptos presentados en este apartado puede representarse en un esquema.

Los diagramas del modelo ER son fáciles de leer, especialmente si uno observa sólo los símbolos gráficos centrales (rectángulos para las entidades, círculos para los atributos, rombos para las interrelaciones, flechas dobles para las jerarquías de generalización, óvalos para los atributos compuestos). La legibilidad decrece si se incluye la cardinalidad de las interrelaciones, la cobertura de las generalizaciones y los identificadores. Para paliar esta dificultad se puede dibujar esquemas ER en diferentes niveles de detalle. En un nivel de abstracción alto se puede incluir sólo entidades, interrelaciones y jerarquías de generalización. Los esquemas correspondientes no incluyen todos los detalles, pero son de lectura fácil. En un nivel de abstracción bajo, se debe incluir todos los rasgos del modelo ER, posiblemente para subconjuntos de la totalidad del esquema. Las conexiones entre los subconjuntos del esquema se determinan en un nivel de abstracción alto.

Para concluir, creemos que el modelo ER representa un buen término medio entre el poder de expresión, la simplicidad y la minimalidad. Muchas críticas al

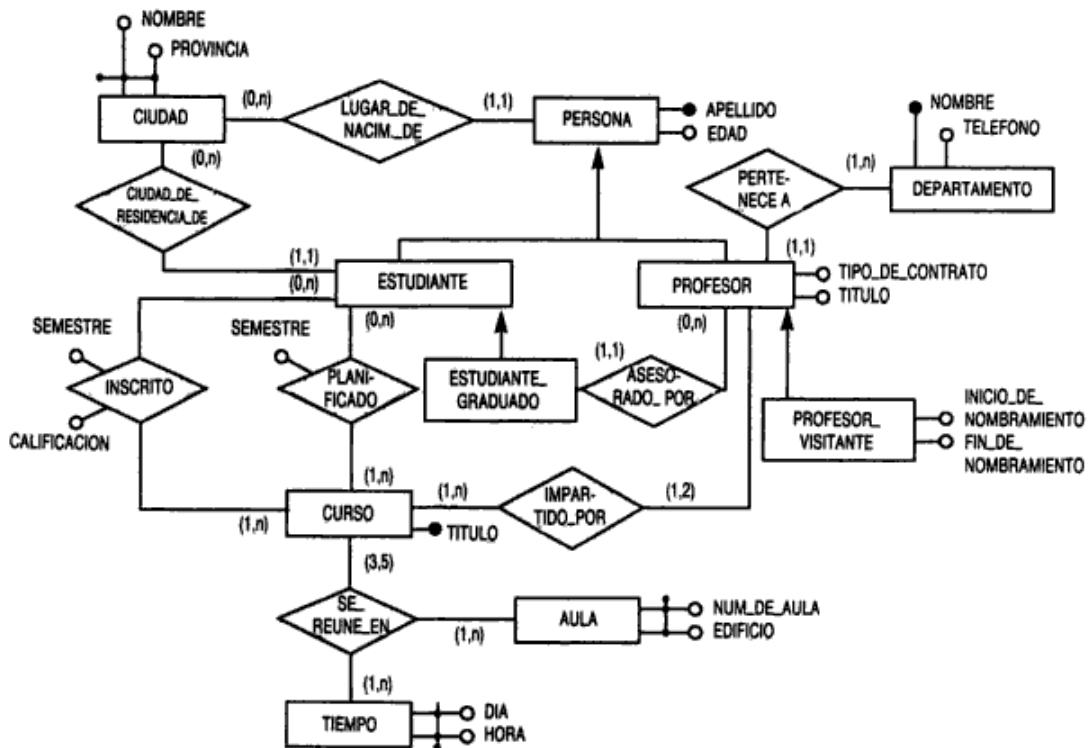


Figura 2.33. Base de datos universitaria.

modelo ER pueden deberse a una presentación pobre de sus cualidades. Se mostrará en los siguientes capítulos metodológicos cómo las diversas características del modelo ER resultan útiles para el diseño de bases de datos. El modelo ER, tal como se usa en el resto del libro, mejora el modelo ER original introducido por Chen en su apoyo para la generalización, los subconjuntos, las restricciones de cardinalidad, el tratamiento de los identificadores, etcétera.

2.5. Lectura de un diagrama de entidades-interrelaciones

El diseño de un esquema conceptual es una actividad compleja que se estudia en los capítulos siguientes. Aquí se propone un ejercicio de lectura de un esquema ER, una habilidad importante que se puede considerar como un primer paso hacia el diseño de uno nuevo. Examíñese el esquema de la figura 2.33, que representa las propiedades de las personas en las universidades. El esquema describe a profesores y a estudiantes; los profesores se relacionan con sus departamentos y lugares de nacimiento, los estudiantes con sus lugares de nacimiento

y de residencia, y con cursos en los que están inscritos; el esquema representa también los lugares de reunión de los cursos y los consejeros de los estudiantes graduados.

Se comienza por concentrar la atención en las generalizaciones y los subconjuntos, que casi siempre representan agregados importantes de información. De acuerdo con sus papeles, las personas se dividen en profesores y estudiantes. Entre los estudiantes se distinguen los estudiantes graduados; entre los profesores se distinguen los visitantes. Luego, se determinan varios agregados de información adicionales:

1. Los profesores se relacionan con su DEPARTAMENTO mediante la interrelación PERTENECE_A; ésta es una interrelación de muchos a uno, ya que cada profesor pertenece a un solo departamento, pero cada departamento tiene varios profesores.
2. Los cursos se relacionan con los profesores que los imparten mediante la interrelación de uno a uno IMPARTIDO_POR; se relacionan con los estudiantes por la interrelación de muchos a muchos INSCRITO y por la interrelación de muchos a muchos PLANIFICADO, que indica el SEMESTRE en que el estudiante piensa tomar el curso.
3. Los lugares de reunión de cada curso se representan por la interrelación ternaria SE_REUNE_EN entre las entidades CURSO, TIEMPO y AULA . La entidad TIEMPO indica el día y hora, y la entidad AULA indica el edificio y número de AULA del lugar de reunión.
4. La interrelación ASESORADO_POR conecta a cada estudiante graduado con su asesor. Se supone que cada estudiante tiene un asesor, y que algunos profesores no son asesores de ningún estudiante; por ello la interrelación ASESORADO_POR es opcional para los profesores, pero obligatoria para los estudiantes graduados.
5. Las ciudades, caracterizadas por su NOMBRE y PROVINCIA, se relacionan con las personas mediante la interrelación LUGAR_DE_NACIMIENTO, y con los estudiantes por la interrelación CIUDAD_DE_RESIDENCIA_DE.

La lectura del esquema se completa al examinar las propiedades de las entidades e interrelaciones individuales. Así, por ejemplo, todas las personas tienen APELLIDO y EDAD, los profesores poseen un TIPO_DE_CONTRATO y un TITULO, los profesores de visita tienen un INICIO_DE_NOMBRAIMIENTO y un FIN_DE_NOMBRAIMIENTO. Los identificadores del esquema son bastante sencillos. Todos son internos; CIUDAD, AULA y TIEMPO tienen identificadores compuestos, mientras que el resto de los identificadores son simples.

2.6. Resumen

Este capítulo ha presentado los conceptos del modelado de datos, incluyendo las abstracciones de datos, los rasgos generales de los modelos conceptuales y el

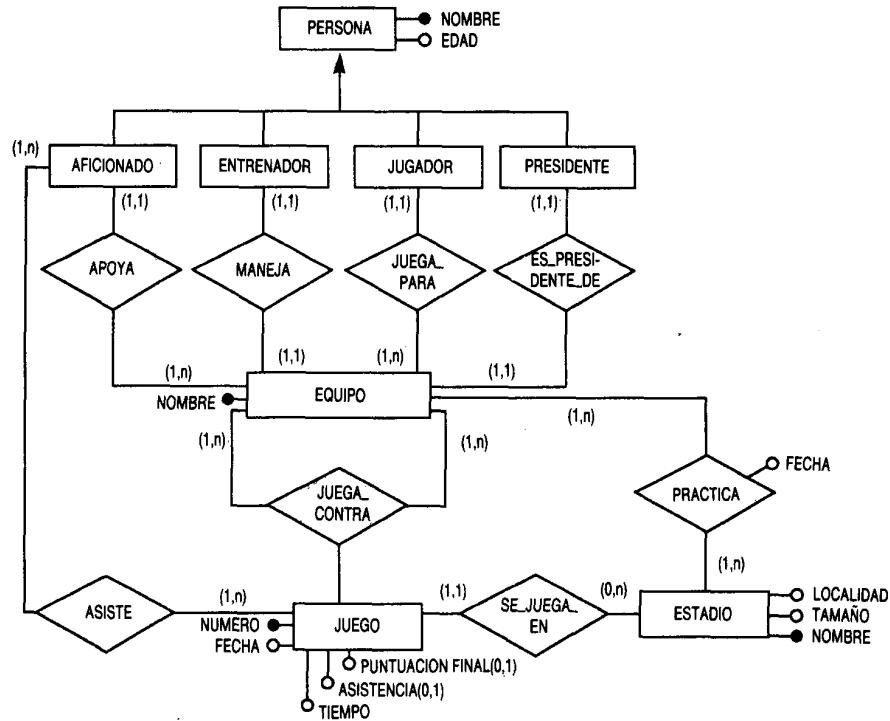


Figura 2.34. Base de datos de fútbol.

modelo de entidades-interrelaciones. En particular, se han introducido los elementos del modelo ER: entidades, interrelaciones, atributos, atributos compuestos, jerarquías de generalización, subconjuntos e identificadores. Este capítulo ha hecho hincapié en la comprensión de un esquema ER existente; con esta base se puede pasar ahora al problema de modelar la realidad en un esquema conceptual.

Ejercicios

- 2.1. Cite al menos cinco ejemplos de cada uno de los tres mecanismos de abstracción analizados en el apartado 2.1.
- 2.2. Elija un tipo de objeto en su habitación (mesa, cama, silla) y muestre cómo se puede representar usando cada uno de los mecanismos de abstracción que se vieron en el apartado 2.1.
- 2.3. Cite algunos ejemplos de abstracciones de generalización y muestre cómo la propiedad de herencia se puede aplicar a sus ejemplos.

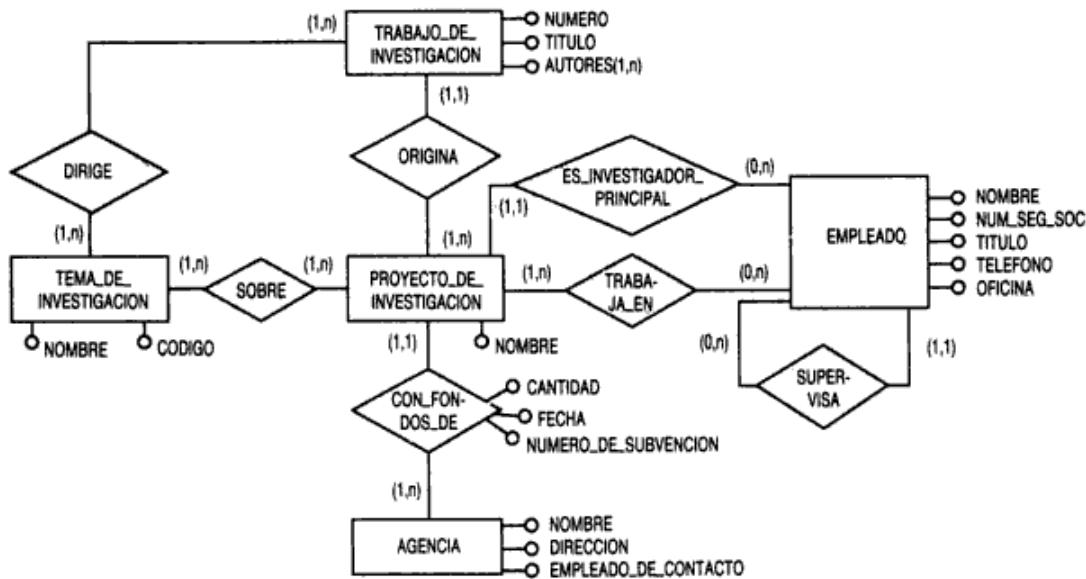


Figura 2.35. Base de datos de proyectos de investigación.

- 2.4. Indique qué mecanismos de abstracción se usan en el esquema ER de la figura 2.20.
- 2.5. Comente cómo las propiedades de cobertura se relacionan con las restricciones de cardinalidad. (*Sugerencia:* Interprete las jerarquías de generalización como tipos especiales de correspondencias entre las clases.)
- 2.6. Especifique lingüísticamente el esquema de la figura 2.33.

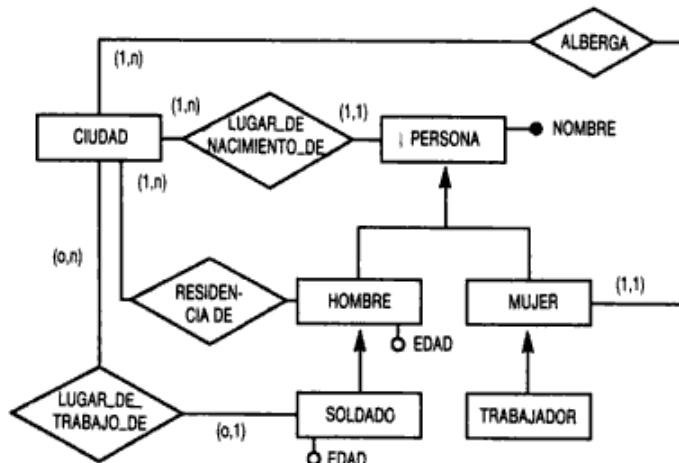


Figura 2.36. Esquema para los ejercicios 2.12 y 2.13.

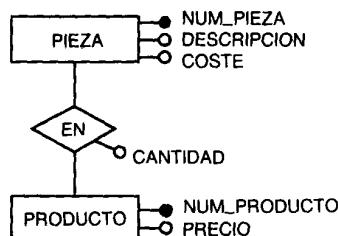


Figura 2.37. Esquema para el ejercicio 2.15.

- 2.7. Lea el esquema ER de la figura 2.34 y dé una descripción narrada de su contenido. Describa todos los rasgos observados.
- 2.8. Lea el esquema ER de la figura 2.35 y dé una descripción narrada de su contenido. Describa todos los rasgos observados.
- 2.9. Proponga identificadores para las entidades del esquema de la figura 2.35.
- 2.10. Cambie el esquema de la figura 2.35 de modo que incluya varias características de los empleados, como sexo, nacionalidad y fecha de nacimiento.
- 2.11. Cambie el esquema de la figura 2.35 de modo que incluya, para empleados específicos que investigan, sus campos de investigación actuales y pasados. ¿Requerirá esto algún identificador adicional en la figura 2.35?
- 2.12. Recuerde la propiedad fundamental de las jerarquías de generalización y simplifique el esquema de la figura 2.36.
- 2.13. Considere de nuevo el esquema de la figura 2.36. ¿Cómo debería modificarlo para representar en el esquema a todos los trabajadores, tanto hombres como mujeres?
- 2.14. Considere el esquema de la figura 2.37, el cual representa los productos de una compañía y las piezas de que están hechos. Indique cómo puede alterarse el esquema para representar lo siguiente:
 - a) Para cada pieza, sus partes componentes.
 - b) Para cada pieza, su color y peso.

Suponga que las piezas están numeradas en serie *dentro de cada producto*, es decir, que una misma pieza puede tener números distintos dentro de distintos productos. ¿Cómo modificaría el esquema de la figura 2.37 para reflejar esta situación? ¿Cuál sería el identificador de PIEZA en este caso?

- 2.15. Como se expone en las notas de la bibliografía, es posible definir nuevas abstracciones y conceptos útiles en tipos específicos de aplicaciones. Examine las abstracciones y modelos propuestos en Brodie (1984), Su (1986), y Di Battista y Batini (1988) y comente cómo pueden representarse en el modelo de entidades-interrelaciones.

Bibliografía

J. R. Abrial, «Data Semantics». En J. W. Klimbie y K. L. Koffeman, eds., *Data Base Management*, North-Holland, 1974, 1-59.

W. Kent, «Limitations of Record-based Information Models», *ACM Transactions on Database Systems*, 4, núm. 1, marzo de 1979, 107-31.

J. M. Smith y D. C. Smith, «Database Abstraction: Aggregation and Generalization», *ACM Transactions on Database Systems*, 2, 1977, 105-33.

D. C. Tsichritzis y F. Lochovsky, *Data Models*, Prentice-Hall, 1982.

La lista anterior incluye las más completas publicaciones sobre modelos de datos. El artículo de Abrial (1974) es históricamente el primero en el área de bases de datos que abordó el problema de definir un modelo verdaderamente *semántico*. Smith y Smith (1977) contribuyeron a la definición formal de las abstracciones. Tsichritzis y Lochovsky (1982) muestran cómo los modelos conceptuales (llamados *modelos infológicos*) pueden considerarse como una evolución de las redes semánticas, originalmente usadas para la representación de conocimientos en el campo de la inteligencia artificial.

M. Brodie, «On the Development of Data Models». En M. Brodie, J. Mylopoulos, y J. Smith, eds., *On Conceptual Modeling*, Springer-Verlag, 1984.

L. Kerschberg, A. Klug, y D. C. Tsichritzis, «A Taxonomy of Data Models». En P. C. Lockemann y E. J. Neuhold, eds., *Systems for Large Data Bases*, North-Holland, 1977, 43-64.

Estos artículos desarrollan taxonomías de modelos de datos; ésta es la base para una muy interesante comparación de las capacidades de modelado de datos.

M. Brodie y D. Ridjanovic, «On the Design and Specification of Database Transactions». En M. Brodie, J. Mylopoulos y J. Schmidt, eds., *On Conceptual Modeling*, Springer-Verlag, 1984.

Este artículo propone una nueva abstracción, la *asociación*, que permite la definición de conceptos nuevos mediante operaciones de agrupación de conceptos existentes (por ejemplo, una familia es una asociación de personas).

P. P. Chen, «The Entity Relationship Model: Toward a Unified View of Data», *ACM Transactions on Database Systems*, 1, núm. 1, marzo de 1976, 9-37.

P. P. Chen, «The Entity-Relationship Model: A Basis for the Enterprise View of Data», *Proceedings IFIPS NCC*, 46, núm. 46, 1977, 76-84.

Los artículos originales sobre el modelo ER. La versión del modelo descrita en estos artículos no incluye jerarquías de generalización ni subconjuntos, e incluye un concepto de entidad débil, que ya mencionamos brevemente.

C. S. Dos Santos, E. J. Neuhold y A. L. Furtado, «A Data Type Approach to the Entity-Relationship Model». En P. Chen, ed., *Entity-Relationship Approach to System Analysis and Design*, North-Holland, 1980, 103-20.

P. Scheuermann, G. Schiffner, y H. Weber, «Abstraction Capabilities and Invariant Properties Modeling within the Entity-Relationship Approach». En P. Chen, ed., *Proc. First International Conference on Entity-Relationship Approach*, Los Angeles, North-Holland, 1980, 121-40.

Los dos artículos anteriores describen varias extensiones del modelo ER que incorporan las jerarquías de generalización y los subconjuntos.

J. Hagelstein y A. Rifaut, «A Semantic Analysis of the Collection Concept». En C. Batini, ed., *Entity-Relationship Approach: A Bridge to the User: Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

L. Tucherman, M. A. Casanova, P. M. Gualandi, y A. P. Braga, «A Proposal for Formalizing and Extending the Generalization and Subset Abstractions in the Entity-Relationship Model». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

Los anteriores artículos ofrecen una descripción clara de la semántica de los conceptos de generalización y colección, y proporcionan un enfoque formal para el estudio de sus propiedades.

P. P. Chen, ed., *Proc. First International Conference on Entity-Relationship Approach*, Los Angeles, North-Holland, 1980.

P. P. Chen, ed., *Proc. Second International Conference on Entity-Relationship Approach*, Washington, D.C., Saugas, Calif.: ER Institute, 1981.

C. Davis et al., eds., *Proc. Third International Conference on Entity-Relationship Approach*, Anaheim, Calif., North-Holland, 1983.

J. Liu, ed., *Fourth International Conference on Entity-Relationship Approach*, Chicago, IEEE Computer Society Press, 1985.

S. Spaccapietra, ed., *Proc. Fifth International Conference on Entity-Relationship Approach*, Dijon, North-Holland, 1986.

S. March, ed., *Proc. Sixth International Conference on Entity-Relationship Approach*, Nueva York, North-Holland, 1987.

C. Batini, ed., *Entity-Relationship Approach: A Bridge to the User: Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

F. Lochovsky, ed., *Proc. Eighth International Conference on Entity -Relationship Approach*, Toronto, North-Holland, 1989.

H. Kangassalo, ed., *Proc. Ninth International Conference on Entity-Relationship Approach*, Lausana, Suiza, North-Holland, 1990.

Las conferencias sobre el enfoque de entidades-interrelaciones se celebraron cada dos años de 1979 a 1985, y después anualmente. Los artículos recopilados en los anales describen el modelado y diseño de bases de datos, no restringidos necesariamente al modelo ER. Los anales de estas conferencias (excepto la cuarta) están disponibles en Estados Unidos y Canadá en texto encuadrado de: North-Holland Publishing Company, c/o Elsevier Science Publishing, Inc., 52 Vanderbilt Avenue, NewYork, NY 10017. En otros lugares debe usarse la dirección que sigue: North-Holland Publishing Company, P.O. Box 1991, 1000 BZ Amsterdam, Países Bajos. Si se desea obtener los anales de la cuarta conferencia, escribase a: IEEE Computer Society, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264 en Estados Unidos.

G. Bracchi, P. Paolini y G. Pelagatti, «Binary Logical Associations in Data Modeling». En G. M. Nijssen, ed. *Modeling in Database Management Systems*, North-Holland, 1979, 125-48.

E. F. Codd, «Extending the Database Relational Model to Capture More Meaning», *ACM Transactions on Database Systems*, 4, núm. 4, 1979.

M. Hammer y D. McLeod, «Database Description with SDM: A Semantic Database Model», *ACM Transactions on Database Systems*, 6, núm. 3, septiembre de 1981.

S. B. Navathe y M. Schkolnick, «View Representation in Logical Database Design», *Proc. ACM-SIGMOD International Conference*, Austin, 1978.

D. W. Shipman, «The Functional Data Model and the Data Language DAPLEX», *ACM Transactions on Database Systems*, 6, núm. 1, 1981.

G. Wiederhold y R. Elmasri, «The Structural Model for Database Design». En P. P. Chen, ed., *The Entity-Relationship Approach to System Analysis and Design*, North-Holland, 1980.

Los artículos anteriores describen varios modelos conceptuales que se pueden considerar alternativos al modelo ER.

S. Abiteboul y R. Hull, «IFO: A Formal Semantic Database Model», *ACM Transactions on Database Systems*, 12, núm. 4, 1987, 525-65.

A. Albano, L. Cardelli, y R. Orsini, «Galileo: A Strongly Typed Interactive Conceptual Language», *ACM Transactions on Database Systems*, 10, núm. 2, junio de 1985.

J. Mylopoulos, P. A. Bernstein, y H. K. Wong, «A Language Facility for Designing Database Intensive Applications», *ACM Transactions on Database Systems*, 5, 1980, 185-207.

Galileo, Taxis e IFO son los modelos semánticos más avanzados propuestos en la literatura. Los consideramos no sólo como herramientas para el diseño de esquemas, sino también (al menos potencialmente) como sistemas de bases de datos completos y lenguajes de programación de bases de datos. Las mejoras con respecto al modelo ER se refieren a (1) un mecanismo fuertemente tipificado para modelar varias propiedades de los conceptos definidos en el modelo, y (2) un uso uniforme de mecanismos de abstracción para modelar datos, operaciones con los datos y sucesos.

R. Hull y R. King, «Semantic Database Modeling: Survey, Applications, and Research Issues», *ACM Computing Surveys*, 19, núm. 3, 1987, 201-60.

J. Peckham y F. Maryanski, «Semantic Data Models», *ACM Computing Surveys* 20, núm. 3, 1988, 153-89.

Estos dos artículos proporcionan una excelente panorámica de los modelos conceptuales más importantes presentados en la literatura y anteriormente mencionados. Los modelos se describen en términos de las abstracciones de modelado subyacentes, constructores de tipos, lenguaje de consulta, primitivas de manipulación de datos y modelado de transacciones.

M. Brodie, J. Mylopoulos, y J. Schmidt, eds., *On Conceptual Modeling*, Springer-Verlag, 1984.

Este libro expone ampliamente la influencia recíproca de la investigación sobre los modelos semánticos, los lenguajes orientados a los objetos y la inteligencia artificial.

G. Kappel y M. Schrefl, «A Behavior Integrated Entity-Relationship Approach for the Design of Object-Oriented Databases». En C. Batini, ed., *Entity-Relationship Approach: A Bridge to the User: Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

R. Lazimy, «E²R Model and Object-Oriented Representation for Data Management, Process Modeling, and Decision Support», En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

S. B. Navathe y M. K. Pillallamarri, «Toward Making the E-R Approach Object Oriented». En C. Batini, ed., *Entity-Relationship Approach: A Bridge to the User: Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

Estos artículos analizan extensiones del modelo ER que incorporan características de orientación a objetos. En un modelo orientado a los objetos los datos no se representan

únicamente en términos de sus propiedades estáticas (por ejemplo, entidades, atributos, restricciones de integridad), sino también en términos de su comportamiento. Un objeto (o tipo de objeto) está, por tanto, formado por un conjunto de propiedades y un conjunto de operaciones; los objetos están normalmente organizados en una jerarquía de generalización por tipo. Los artículos describen varios modelos ER provistos de extensiones orientadas a objetos y sus correspondientes metodologías de diseño.

G. Di Battista y C. Batini, «Design of Statistical Databases: A Methodology for the Conceptual Step», *Information Systems*, 13, núm. 4, 1988, 407-22.

S. Y. W. Su, «Modeling Integrated Manufacturing Data with SAM*», *IEEE Computer*, enero de 1986, 34-49.

Estos artículos analizan los modelos conceptuales para aplicaciones estadísticas. En general, las aplicaciones estadísticas necesitan modelar las llamadas agregaciones complejas; por ejemplo, en un censo de población, una agregación compleja es la estatura media de las personas agrupadas por edad y ciudad de nacimiento. En este ejemplo, estamos interesados en una característica de las personas (estatura promedio) agrupadas según todos los posibles tipos de edad y ciudades de nacimiento. Tales agregaciones no se modelan fácilmente en el modelo ER; por esta razón, en estos artículos el modelo ER es ampliado con conceptos estadísticos, como agregaciones estadísticas, clases de objetos (PERSONAS en el ejemplo anterior), atributos de categoría (EDAD y CIUDAD_DE_NACIMIENTO), y atributos de resumen (EDAD promedio).

R. Goldstein y V. Storey, «Some Findings on the Intuitiveness of Entity-Relationship Constructs». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

Este artículo describe una experiencia extensa de capacitación de usuarios finales en el modelo de entidades-interrelaciones. Contiene varias construcciones incorrectas típicas y se analizan las causas de las discrepancias.

D. Batra, J. Hoffer, y P. Bostrom, «Comparing Representations with Relational and ER Models», *Communications of the ACM*, 33, núm. 2, febrero de 1990, 126-40.

Este artículo trata la facilidad de uso del modelo conceptual por parte de usuarios neófitos, comparando la eficacia del modelo entidades-interrelaciones con respecto al modelo relacional. Propone un esquema de calificación para medir la corrección en el uso del modelo. El resultado muestra un mejor rendimiento del modelo entidades-interrelaciones.

S. Dart, R. Ellison, P. Feifler, y N. Habermann, «Software Development Environments», *IEEE Computer*, 20, núm. 11, 18-28.

Este artículo analiza varios tipos de entornos utilizados por diseñadores de sistemas para construir sistemas de software. Los entornos centrados en el lenguaje están construidos alrededor de un lenguaje, y proporcionan un conjunto de herramientas apropiadas para ese lenguaje. Los entornos centrados en la estructura incorporan técnicas independientes del lenguaje que permiten al usuario manipular directamente las estructuras. Los entornos de juegos de herramientas suministran una colección de herramientas que incluyen un apoyo independiente del lenguaje para grandes tareas de programación, como la administración de configuraciones y el control de versiones. Finalmente, los entornos basados en el método apoyan una amplia gama de actividades en los procesos de desa-

rrollo de software e incorporan herramientas para especificaciones y métodos de diseño particulares. El enfoque de entidades-interrelaciones se considera un entorno basado en el método; se clasifica como un método semiformal porque la definición es intuitiva y usa descripciones gráficas.

Metodología para el diseño conceptual

La creación de un esquema ER es un proceso incremental: la percepción de la realidad se refina y enriquece de forma progresiva, y el esquema conceptual se desarrolla gradualmente. El proceso puede hacerse más fluido por el uso de transformaciones estructuradas. En este capítulo se presentan las *primitivas de refinamiento*, un conjunto limitado de transformaciones que se aplican a un esquema inicial para crear un esquema final. Las primitivas se clasifican como descendentes o ascendentes, con base en las características de los esquemas inicial y final. Una aplicación rigurosa de las primitivas ayuda a los diseñadores, especialmente al inicio, por ser fáciles de aplicar y por dividir el proceso de diseño en pasos simples y seguros.

Se desarrollarán varias *estrategias de diseño* al usar sólo ciertos tipos de primitivas de refinamiento. Una estrategia *descendente* pura refina los conceptos abstractos hasta hacerlos concretos; una estrategia *ascendente* pura sigue el camino contrario. Estos enfoques representan dos extremos. Otras estrategias menos radicales incluyen la estrategia *mixta* y la *centrífuga* (del centro hacia afuera).

Las primitivas y las estrategias son los pilares sobre los que se construyen las *metodologías de diseño*. Una buena metodología para el diseño conceptual debe ser, idealmente, un término medio entre dos aspectos contrastantes. Por un lado, la metodología debe ser *rigurosa*; ha de sugerir una estrategia para todas las decisiones importantes que deben tomarse en el proceso de diseño. Una estrategia de estas características debe basarse en un enfoque formal y cada toma de decisión debe corresponder idealmente a un algoritmo. Por otro lado, la metodología debe ser *flexible*, aplicable a una diversidad de situaciones y entornos. En consecuencia, la metodología debe tener suficiente flexibilidad para que cada diseñador sea capaz de adaptarla a las restricciones específicas de una organización y pueda seguir su propio estilo de diseño.

Este libro se acomoda a las dos necesidades al proporcionar un único marco metodológico consistente en una definición clara de las fases de diseño, sus objetivos y las técnicas que requiere cada fase. Al mismo tiempo, se ofrece una variedad de estrategias para adaptar esas técnicas a situaciones de diseño específicas. El apartado 3.1 presenta las primitivas de refinamiento, y el 3.2, las es-

trategias de diseño. Estos son los pilares que sostienen el desarrollo de una metodología de diseño. En el apartado 3.3 se da un primer grupo de reglas empíricas para elegir conceptos (entidades, interrelaciones, etc.) al modelar los requerimientos. El apartado 3.4 presenta una visión metodológica global del proceso de diseño conceptual al describir entradas, salidas y principales actividades de dicho proceso.

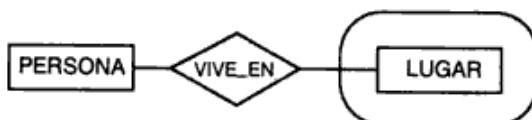
3.1. Primitivas del diseño conceptual

El diseño de un esquema conceptual es el resultado de un análisis complejo de los requerimientos del usuario. Como consecuencia, el esquema comúnmente es producto de un proceso iterativo. Durante ese proceso, se empieza con una versión preliminar del esquema y se efectúa una serie de *transformaciones de esquemas* que finalmente producen la versión definitiva. Es una situación similar a la que se presenta en el diseño de software, donde el programa final suele producirse desarrollando un conjunto de transformaciones que enriquecen, en una secuencia de pasos, una primera versión del programa. En el diseño de datos se define un conjunto de primitivas de refinamiento que pueden usarse para efectuar transformaciones, cada una de las cuales se aplica a un esquema inicial y produce un esquema resultante.

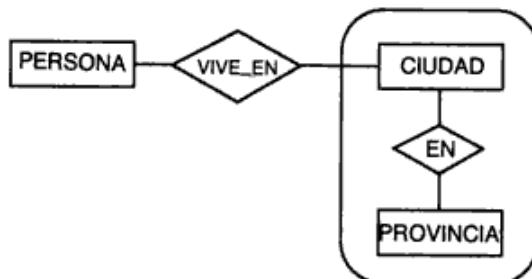
Considérense los dos esquemas de la figura 3.1. El esquema resultante (Fig. 3.1b) se obtiene del esquema inicial (Fig. 3.1a) mediante una transformación (Fig. 3.1c). El propósito de la transformación es refinar el concepto abstracto LUGAR en términos de conceptos más específicos: las dos entidades CIUDAD y PROVINCIA y la interrelación binaria EN que se define entre ellas. La interrelación VIVE_EN, definida originalmente entre PERSONA y LUGAR, se conecta a la entidad CIUDAD.

Las transformaciones de esquemas tienen tres características bien definidas:

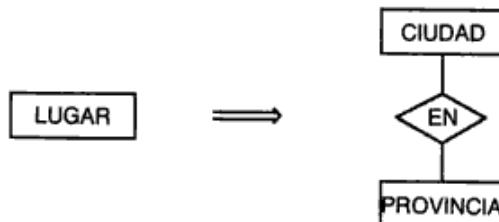
1. Cada transformación de esquema tiene un **esquema inicial**, esto es, el esquema al que se aplica la transformación, y un **esquema resultante**, el efecto de aplicar la transformación. En la figura 3.1, el esquema inicial es una entidad y el esquema resultante es un par de entidades conectadas por una interrelación.
2. Cada transformación de esquema crea una correspondencia entre **nombres** de conceptos del esquema inicial y nombres de conceptos del esquema resultante. En la figura 3.1, la correspondencia de nombres se da entre LUGAR y los nuevos nombres CIUDAD, EN y PROVINCIA.
3. Los conceptos del esquema resultante deben heredar todas las conexiones lógicas definidas para los conceptos del esquema inicial. Por ejemplo, las entidades están conectadas dentro de un esquema de entidades-interrelaciones con interrelaciones, atributos y otras entidades en generalizaciones y subconjuntos. Cuando una entidad se transforma en un conjunto de conceptos, esos



(a) Esquema inicial



(b) Esquema resultante



(c) Transformación

Figura 3.1. Ejemplo de transformación de esquemas.

conceptos deben heredar todas las conexiones de la entidad en el esquema anterior. En la figura 3.1, el nexo lógico entre PERSONA y LUGAR, dado por la interrelación VIVE_EN, lo hereda la entidad CIUDAD.

De la misma manera que en el diseño de software, los tipos de transformaciones utilizados en el proceso de diseño influyen fuertemente en la calidad global de la actividad de diseño. Este hecho justifica un análisis profundo de las transformaciones y sus propiedades: el propósito final es encontrar un pequeño conjunto de transformaciones que se puedan utilizar con eficacia en el modelado de requerimientos. Por esta razón, la exposición se restringe a un conjunto limitado de **transformaciones primitivas**, es decir, transformaciones con una estructura simple, que no se puedan descomponer en otras más simples.

Las primitivas se clasifican en dos grupos: descendentes y ascendentes. Las primitivas descendentes corresponden a refinamientos **puros**, es decir, refina-

mientos que se aplican a un concepto simple (el esquema inicial) y producen una descripción más detallada de ese concepto (el esquema resultante). En cambio, las primitivas ascendentes introducen conceptos *nuevos* y propiedades que no aparecían en versiones anteriores del esquema.

3.1.1. Primitivas descendentes

Las primitivas descendentes se caracterizan por las siguientes propiedades:

1. Tienen una estructura simple: el esquema inicial es un concepto único y el esquema resultante se compone de un pequeño conjunto de conceptos.
2. Todos los nombres se refinan dando lugar a nuevos nombres que describen el concepto original en un nivel de abstracción más bajo.
3. Las conexiones lógicas deben ser heredadas por un solo concepto del esquema resultante.

La primitiva mostrada en la figura 3.1 es descendente: se aplica a un concepto único, usa nombres nuevos en el esquema final, y la interrelación VIVE_EN se aplica a un solo concepto del esquema resultante, la entidad CIUDAD.

Las primitivas descendentes se presentan y clasifican en la figura 3.2 y se ejemplifican en la figura 3.3. Cada primitiva realiza un tipo específico de refinamiento del esquema.

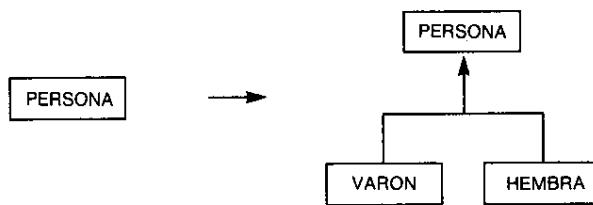
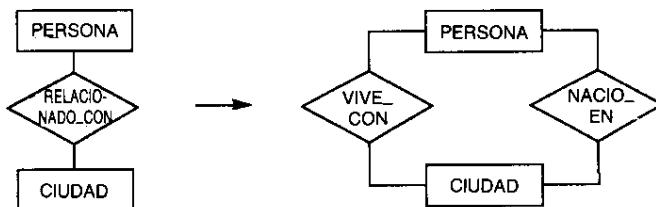
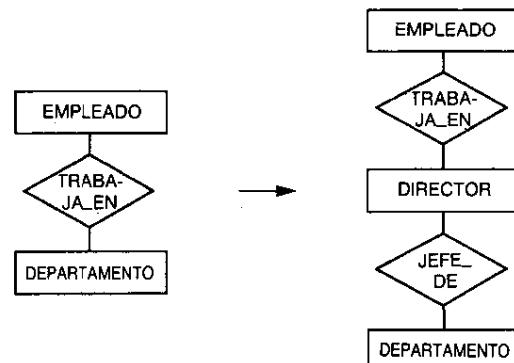
1. La primitiva T_1 refina una entidad para producir una interrelación entre dos o más entidades. El ejemplo de la figura 3.1 es una aplicación de esta primitiva.
2. La primitiva T_2 refina una entidad para producir una jerarquía de generalización o un subconjunto. La figura 3.3a muestra una aplicación de esta primitiva: la entidad PERSONA se refina en una generalización que incluye VARON y HEMBRA.
3. La primitiva T_3 divide una entidad en un conjunto de entidades independientes. El efecto de esta primitiva es introducir nuevas entidades, no establecer interrelaciones o generalizaciones entre ellas. En la figura 3.3b, la entidad PREMIO se divide en dos entidades, PREMIO_NOBEL y OSCAR. No se establece ninguna interrelación entre las dos entidades, pues son dos formas distintas e independientes de clasificar los premios.
4. La primitiva T_4 refina una interrelación para producir dos (o más) interrelaciones de las mismas entidades. En la figura 3.3c, la interrelación RELACIONADO_CON, entre personas y ciudades, se refina para dar las dos interrelaciones VIVE_EN y NACIO_EN.
5. La primitiva T_5 refina una interrelación para producir una *ruta* de entidades y interrelaciones. La aplicación de esta primitiva comprende el reconocimiento de que una interrelación entre dos conceptos debe expresarse mediante un tercer concepto, que ha estado oculto en la representación ante-

Primitiva	Esquema inicial	Esquema resultante
T ₁ : Entidad→Entidades relacionadas		
T ₂ : Entidad→Generalización; (Entidad→Subconjunto)		
T ₃ : Entidad→Entidades no relacionadas		
T ₄ : Interrelación→Interrelaciones paralelas		
T ₅ : Interrelación→Entidad con interrelaciones		
T ₆ : Desarrollo de atributos		
T ₇ : Desarrollo de atributos compuestos		
T ₈ : Refinamiento de atributos		

Figura 3.2. Clasificación de las primitivas descendentes.

rior. En la figura 3.3d se refina la interrelación TRABAJA_EN entre EMPLEADO y DEPARTAMENTO para dar una agregación más compleja que incluye la entidad DIRECTOR y dos interrelaciones nuevas.

6. La primitiva T₆ refina una entidad (o una interrelación) introduciendo sus atributos. En la figura 3.3e se crean los atributos NOMBRE, SEXO y EDAD para la entidad PERSONA.

(a) Aplicación de la primitiva T_2 (b) Aplicación de la primitiva T_3 (c) Aplicación de la primitiva T_4 (d) Aplicación de la primitiva T_5 (e) Aplicación de la primitiva T_6 **Figura 3.3.** Ejemplos de aplicaciones de primitivas descendentes.

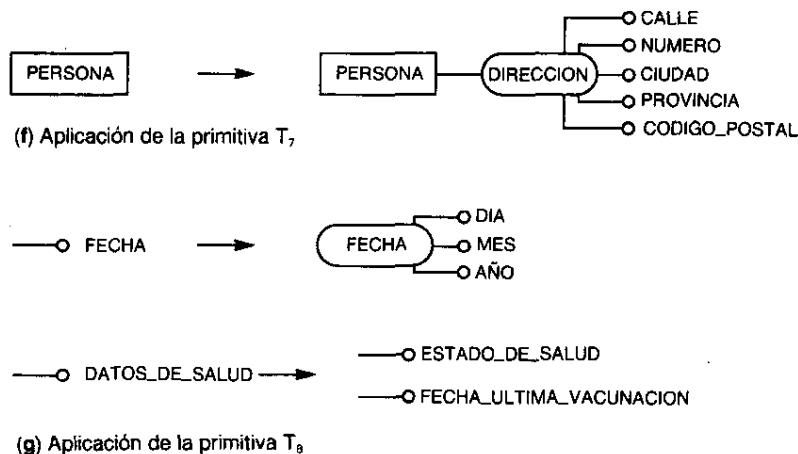
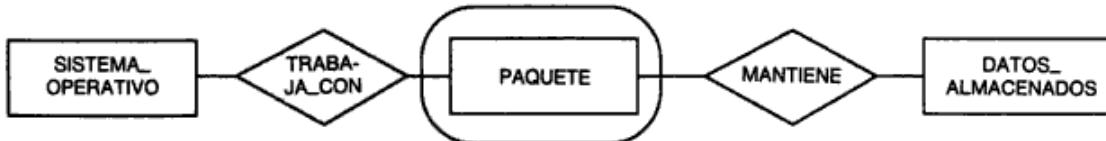


Figura 3.3.Bis Ejemplos de aplicaciones de primitivas descendentes (*continuación*).

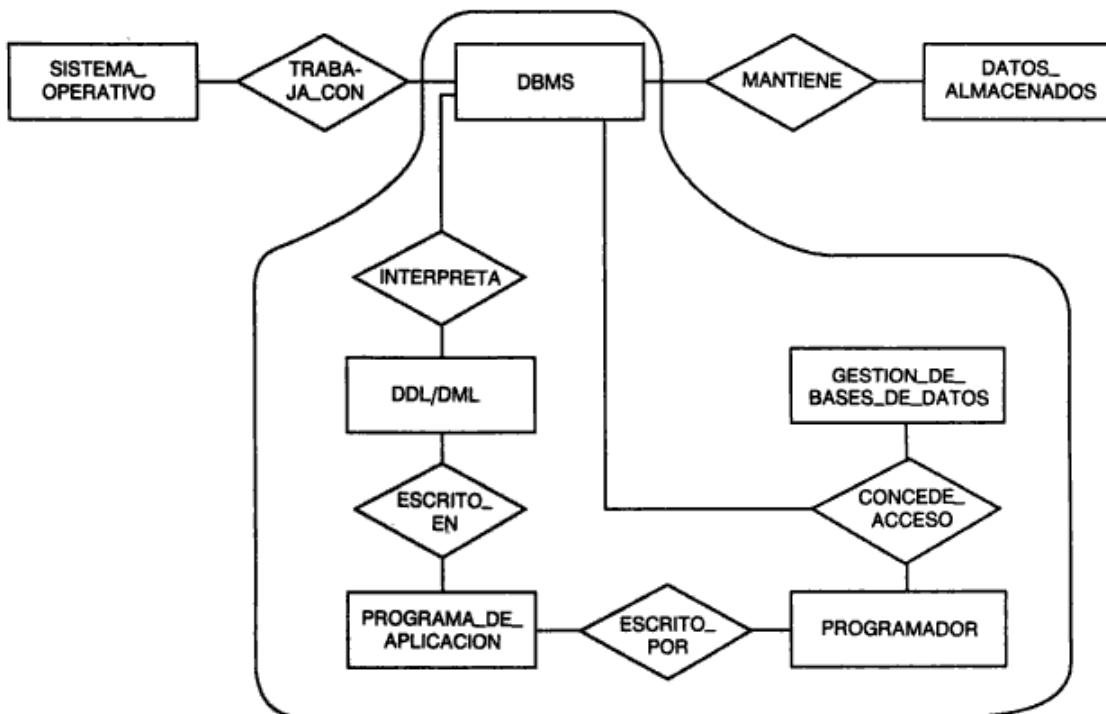
7. La primitiva T_7 refina una entidad (o una interrelación) introduciendo un atributo compuesto. En la figura 3.3f se crea el atributo compuesto DIRECCION para la entidad PERSONA.
8. La primitiva T_8 refina un atributo simple para producir un atributo compuesto o bien un grupo de atributos. En la figura 3.3g, el atributo FECHA se refina para dar un atributo compuesto con tres atributos: DIA, MES y AÑO. El atributo DATOS_DE_SALUD se refina en términos de los atributos ESTADO_DE_SALUD y ULTIMA_VACUNACION.

Las primitivas anteriores tienen esquemas resultantes simples y pueden considerarse como las herramientas más sencillas para refinar un esquema. Su uso debe maximizarse para lograr un proceso de diseño estructurado, comprensible y fiable. Otras transformaciones de esquemas, más complejas, pueden clasificarse como descendentes. La figura 3.4 muestra una transformación de esquemas descendente compleja que refina una entidad para dar un conjunto de entidades e interrelaciones. El esquema se refiere a una parte del entorno de un DBMS.

Al aplicar las primitivas de refinamiento se debe respetar, implícitamente, ciertas restricciones. Considérese como caso la primitiva T_5 ; existe un vínculo claro entre las cardinalidades mínima y máxima de la interrelación en el esquema inicial y en el esquema resultante. Por ejemplo, si el esquema inicial posee una interrelación de uno a uno, el esquema resultante no puede incluir dos interrelaciones de muchos a muchos. Estas restricciones, empero, dependen también del significado de los conceptos sometidos a refinamiento. Considérense, por ejemplo, las dos aplicaciones diferentes de la primitiva T_4 , mostradas en la figura 3.5. En la primera situación, la interrelación RELACIONADO_CON es un modelado preliminar de dos interrelaciones distintas e independientes, pro-



(a) Esquema inicial



(b) Esquema resultante

Figura 3.4. Aplicación de una transformación descendente compleja a un esquema.

ducidas por el refinamiento; en este caso no existe restricción sobre las cardinales mínima y máxima de las dos interrelaciones. En la segunda situación, los casos de las interrelaciones DIRIGE y TRABAJA_EN constituyen una partición de los casos de la interrelación EMPLEADO_EN.

3.1.2. Primitivas ascendentes

Las primitivas ascendentes introducen nuevos conceptos y propiedades que no aparecían en versiones anteriores del esquema, o modifican algunos conceptos

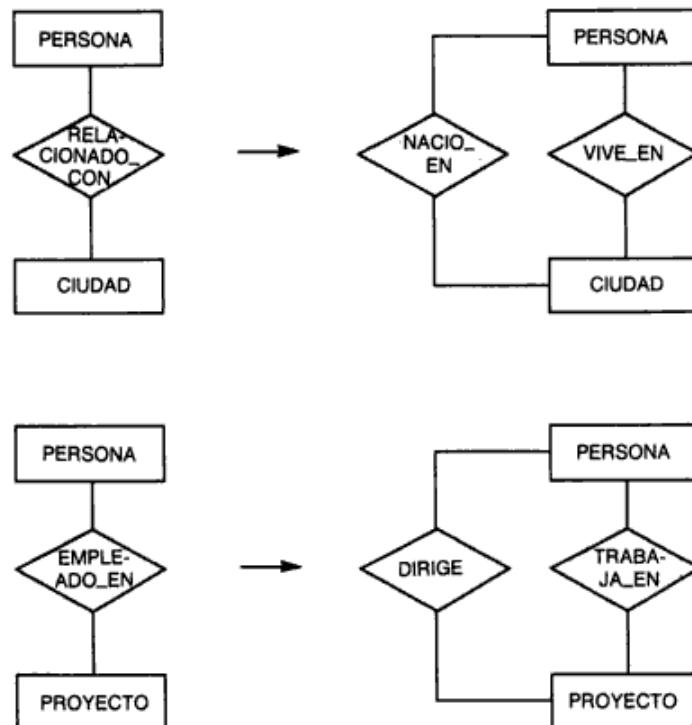


Figura 3.5. Dos aplicaciones de la primitiva T_4 .

existentes. Las primitivas ascendentes se usan en el diseño de un esquema siempre que se descubren rasgos del dominio de aplicación que no fueron captados en ningún nivel de abstracción en la versión anterior del esquema. Las primitivas ascendentes se aplican, asimismo, cuando se fusionan esquemas diferentes para formar un esquema global más amplio. Esta acción, llamada *integración*, se trata en el capítulo 5.

Las primitivas ascendentes más utilizadas en el curso de un diseño se clasifican en la figura 3.6 y se exemplifican en la figura 3.7.

1. La primitiva B_1 genera una nueva entidad. Esta primitiva se usa cuando el diseñador descubre un nuevo concepto con propiedades específicas, que no aparecía en el esquema anterior.
2. La primitiva B_2 genera una nueva interrelación entre entidades definidas anteriormente. En la figura 3.7a, una nueva interrelación *VIVE_EN* se establece entre PERSONA y LUGAR.
3. La primitiva B_3 crea una nueva entidad que se toma como una generalización (sea un subconjunto o una jerarquía de generalización) entre entidades definidas con anterioridad. En la figura 3.7b se genera una jerarquía de generalización, creando una nueva entidad, PERSONA.

Primitiva	Esquema inicial	Esquema resultante
B ₁ : Generación de entidad		
B ₂ : Generación de interrelación		
B ₃ : Generación de generalización (generación de subconjuntos)		
B ₄ : Agregación de atributos		
B ₅ : Agregación de atributo compuesto		

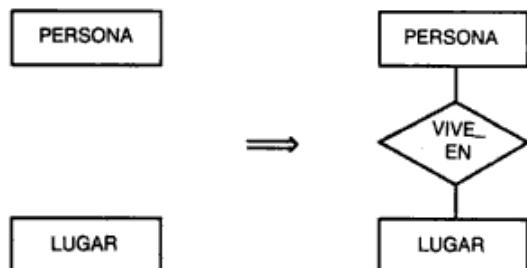
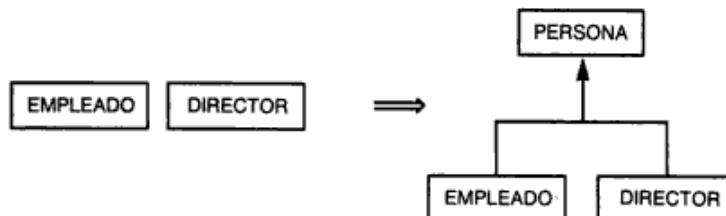
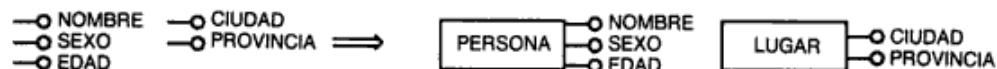
Figura 3.6. Clasificación de las primitivas ascendentes.

4. La primitiva B₄ genera un nuevo atributo y lo conecta a una entidad o interrelación definida anteriormente. En la figura 3.7c, los atributos NOMBRE, SEXO y EDAD se añaden a la entidad PERSONA.
5. La primitiva B₅ crea un atributo compuesto y lo conecta a una entidad o interrelación ya definida. En la figura 3.7d, el atributo compuesto DIRECCIÓN se añade a la entidad PERSONA.

La aplicación de la primitiva ascendente B₃ hace necesario comprobar si las propiedades deben pasar de una entidad a otra, debido a la nueva generalización. En la figura 3.8 se introducen una nueva entidad, PERSONA, y un subconjunto entre PERSONA y EMPLEADO, aplicando la primitiva B₃; como consecuencia, los atributos NOMBRE_DE_PILA y APELLIDO y la interrelación NACIDO_EN deben subir de posición y ser asignados a la entidad genérica. Este ejemplo muestra que las primitivas ascendentes pueden forzar a los diseñadores a investigar las consecuencias de la transformación en el resto del esquema.

3.1.3. Propiedades de las primitivas

En este apartado se investigan las propiedades de las primitivas descendentes y ascendentes. En primer lugar es necesario definir las cualidades de **compleción** y **minimalidad** para un conjunto genérico de primitivas. Un conjunto de primitivas es completo si cualquier esquema de base de datos puede construirse a partir de un esquema inicial vacío, mediante la aplicación de una secuencia de primitivas del conjunto. Un conjunto de primitivas es mínimo si ninguna primitiva del conjunto puede expresarse usando las otras primitivas.

(a) Aplicación de la primitiva B_2 (b) Aplicación de la primitiva B_3 (c) Aplicación de la primitiva B_4 (d) Aplicación de la primitiva B_5 **Figura 3.7.** Ejemplos de aplicaciones de las primitivas ascendentes.

Cuando se considera la compleción y la minimalidad respecto a las primitivas descendentes y ascendentes, se ve que *no todas* las primitivas descendentes son mínimas. La primitiva T_5 , por ejemplo, puede expresarse en términos de la primitiva T_1 . Por otro lado, el conjunto $\{T_1, T_2, T_3, T_4, T_6, T_7\}$ es mínimo: T_1 se necesita para generar interrelaciones, T_2 para generar generalizaciones, T_3 para generar esquemas desconectados, T_6 para generar atributos y T_7 para generar atributos compuestos. Para demostrar la minimalidad de T_4 , nótese que T_1 sólo puede generar árboles de interrelaciones, no ciclos. T_4 puede generar ciclos; por

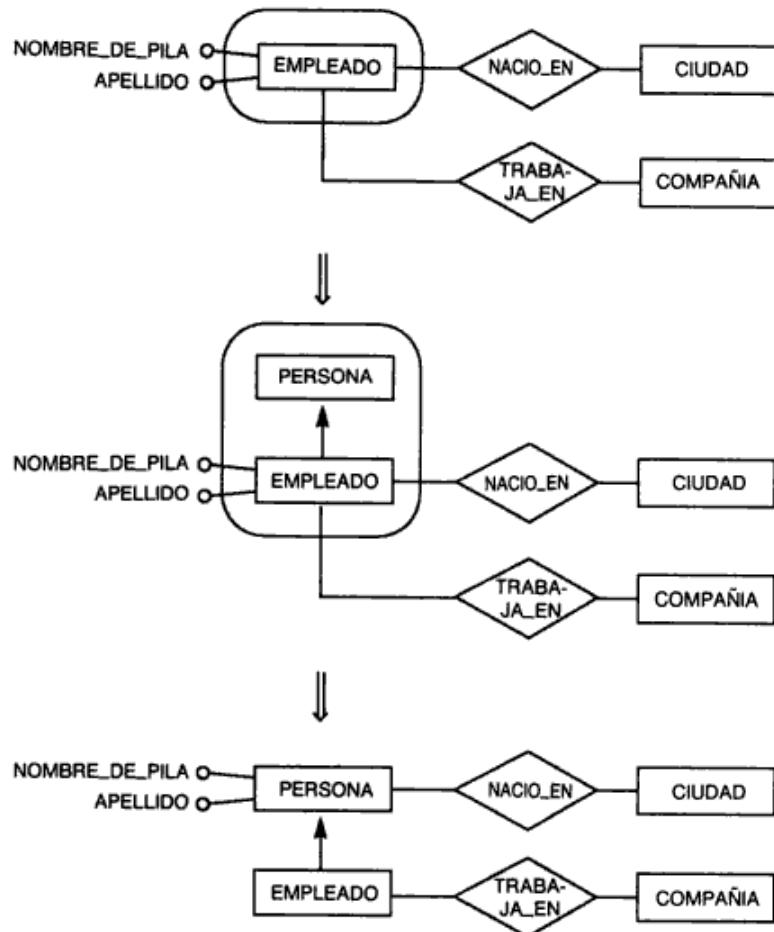


Figura 3.8. Migración de propiedades después de la aplicación de una primitiva ascendente.

tanto, T_4 permite la generación de esquemas que no pueden ser generados por otras primitivas.

Las primitivas descendentes *tampoco* son completas. Para demostrarlo basta con encontrar esquemas que no se puedan generar con sólo usar las primitivas descendentes. En efecto, los gráficos completos de interrelaciones no pueden generarse mediante primitivas descendentes. De hecho, sólo permiten generar gráficos que se pueden reducir a conexiones en serie o en paralelo de interrelaciones. La figura 3.9 muestra un gráfico completo que no puede generarse sólo con primitivas descendentes.

En cambio, las primitivas ascendentes son mínimas: cada primitiva introduce un concepto distinto del modelo. Las primitivas ascendentes son también completas. Todo esquema puede generarse introduciendo primero entidades e

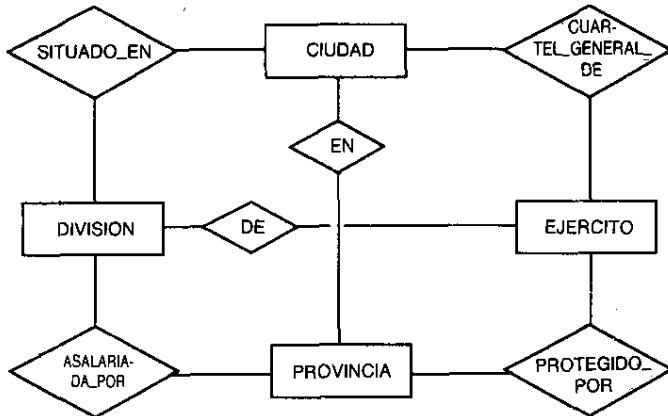


Figura 3.9. Ejemplo de un esquema que no puede generarse por la aplicación de primitivas descendentes.

insertando luego el resto de los conceptos relacionados usando las primitivas apropiadas.

Las propiedades de los dos tipos de primitivas permiten concluir que todos los esquemas pueden generarse de forma ascendente, pero que sólo algunos esquemas pueden generarse de forma descendente; estos últimos se llaman esquemas *producibles en forma descendente*.

3.2. Estrategias para el diseño de esquemas

Se distinguen cuatro estrategias para el diseño de esquemas: descendente, ascendente, centrifuga y mixta. Cada una se caracteriza por el uso de tipos particulares de primitivas. En el ejemplo que sigue se aplican estas estrategias en el diseño de una base de datos demográfica que representa varias propiedades de las personas y lugares geográficos. Los requerimientos para la base de datos son los siguientes:

- En la base de datos de un censo se consideran las siguientes propiedades de las personas: nombre, apellido, sexo, edad, lugar de nacimiento, lugar de residencia, años de residencia, situación militar de los hombres, apellido de soltera de las mujeres.
- Los lugares pueden ser estados extranjeros o ciudades nacionales. Cada uno tiene un nombre y número de habitantes (que representa la población total en el caso de los estados extranjeros) y los nombres de las regiones o ciudades.

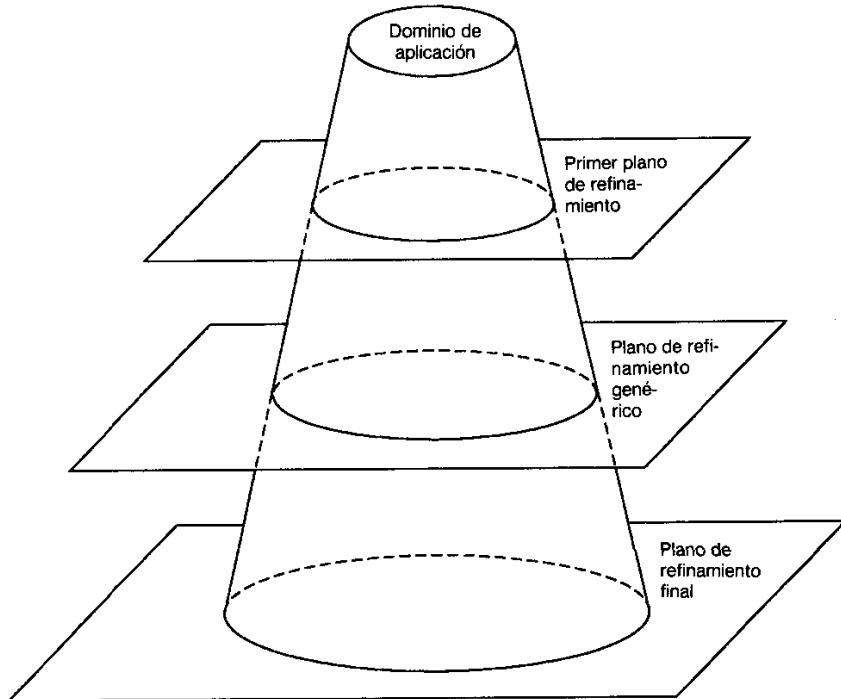


Figura 3.10. La estrategia descendente.

3.2.1. Estrategia descendente

En la estrategia descendente se obtiene un esquema aplicando sólo las primitivas de refinamiento descendente; cada primitiva introduce nuevos detalles en el esquema. El proceso termina cuando están representados todos los requerimientos. La figura 3.10 muestra una representación abstracta en forma de cono del proceso de diseño: en cada transformación descendente el diseñador se mueve de un plano de diseño a otro, mientras que la parte del dominio de aplicación representada por el cono se mantiene inalterable. Es importante señalar que, en una estrategia descendente «pura», todos los conceptos que se representan en el esquema final deben estar presentes en cada plano de refinamiento.

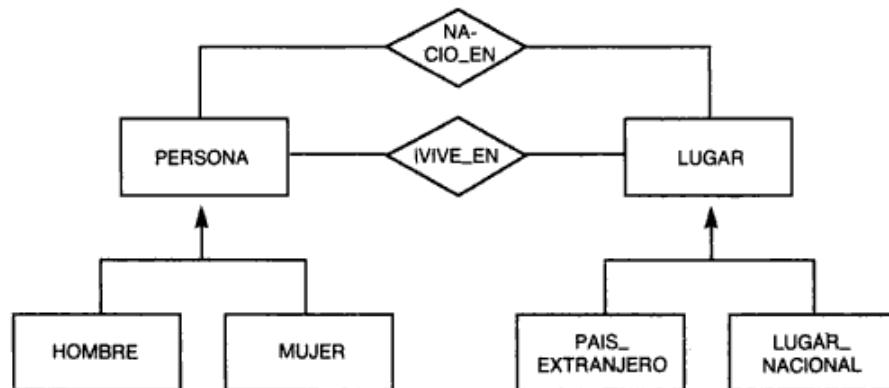
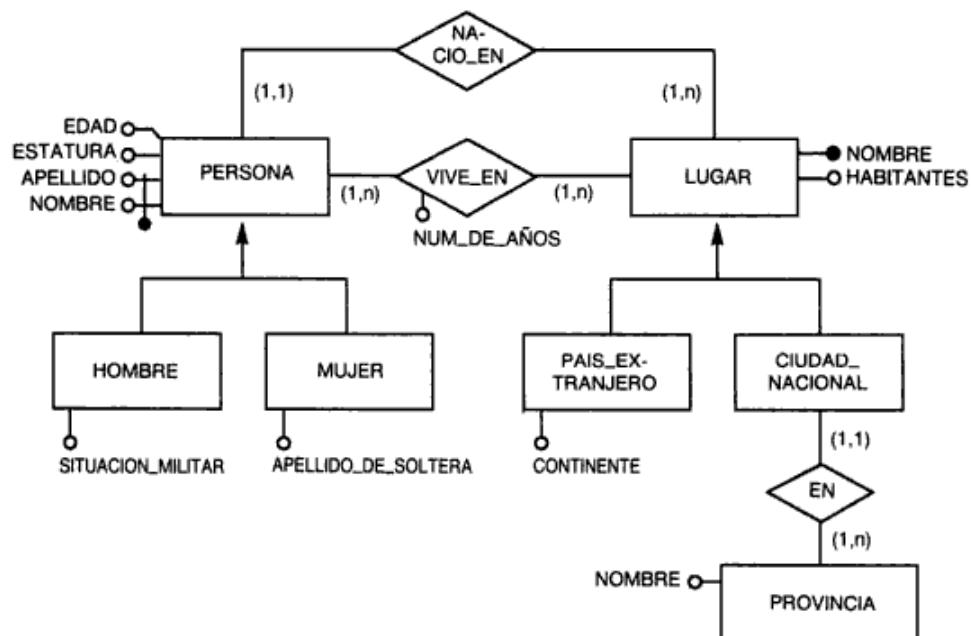
La figura 3.11 muestra el diseño de la base de datos demográfica usando un enfoque descendente puro. El primer esquema representa sólo un concepto, la entidad **DATOS_DEMOGRAFICOS**, que modela toda la aplicación de forma muy abstracta. El segundo esquema refina esa entidad para dar dos entidades, **DATOS_DE_PERSONAS** y **DATOS_DE_LUGARES**, y una interrelación entre ellas. Esto representa explícitamente el hecho de que los datos se refieren a dos tipos de objetos de la realidad, personas y lugares. En el tercer esquema se desarrollan



(a) Primer refinamiento



(b) Segundo refinamiento

(c) Tercer refinamiento (primitivas T₂ y T₄)(d) Esquema final (primitivas T₁, T₂, y T₆)**Figura 3.11.** Una sesión de diseño usando la estrategia descendente.

tres refinamientos distintos: 1) se distinguen dos interrelaciones diferentes entre las personas y los lugares: NACIO_EN y VIVE_EN; 2) se introducen dos tipos diferentes de personas, representados por las entidades HOMBRE y MUJER; 3) se distinguen dos tipos diferentes de lugares: PAIS_EXTRANJERO y LUGAR_NACIONAL.

El cuarto esquema introduce los detalles finales: 1) se especifican los atributos; 2) la entidad LUGAR_NACIONAL se refina para dar dos entidades, CIUDAD_NACIONAL y provincia, y una nueva interrelación se define entre ellas; asimismo, un nuevo atributo, NOMBRE, se define para PROVINCIA; 3) se especifican las cardinalidades de las interrelaciones; 4) se especifican los identificadores.

Al emplear refinamientos puros e independientes, el diseñador es capaz de analizar un concepto a la vez, ignorando otros detalles, con lo que se simplifica el proceso iterativo. Nótese, empero, que el enfoque descendente sólo es aplicable cuando el diseñador es capaz de construir *en su mente* una vista de alto nivel, que abarca todos los requerimientos. Esto suele ser muy difícil, especialmente en el caso de bases de datos grandes.

3.2.2. Estrategia ascendente

Con la estrategia ascendente se obtiene un esquema aplicando sólo las primitivas de refinamiento ascendentes, partiendo de conceptos elementales y construyendo conceptos más complejos a partir de ellos; los requerimientos se descomponen, se conceptualizan de manera independiente y, finalmente, se fusionan en un esquema global. La figura 3.12 muestra este proceso: comenzamos por producir, en cualquier orden, los conceptos elementales pertenecientes al esquema; después, usamos las estructuras provistas por el modelo conceptual para agregarlas progresivamente hasta el esquema final.

Ahora se diseña el esquema demográfico usando un estilo ascendente puro (Fig. 3.13). El primer producto de este diseño es el conjunto de todas las propiedades atómicas, representadas por los atributos. El primer esquema no tiene estructura: carece de mecanismos de abstracción, excepto los atributos, que pueden verse como abstracciones de dominios. En el segundo esquema, obtenido mediante la primitiva B₄, se superpone una primera estructura de abstracción sobre los atributos, al definir entidades que representan abstracciones de agregación de los atributos. En el tercer esquema, obtenido mediante la primitiva B₃, se superponen las abstracciones de generalización (jerarquías de generalización y subconjuntos) sobre las entidades. Obsérvese que la progresiva adición de abstracciones puede causar una reestructuración parcial del esquema, debido a la migración de propiedades a lo largo de las generalizaciones. Por último, en el cuarto esquema, obtenido mediante la primitiva B₂, se añaden interrelaciones entre las entidades y se introducen las cardinalidades y los identificadores. La sesión de diseño propuesta representa un caso extremo de la estrategia ascendente, donde los diversos tipos de abstracciones se introducen progresivamente. Con un enfoque ascendente diferente, es posible introducir

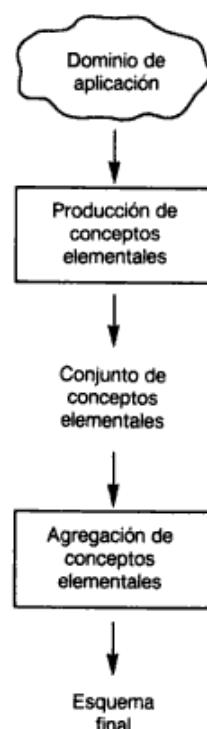
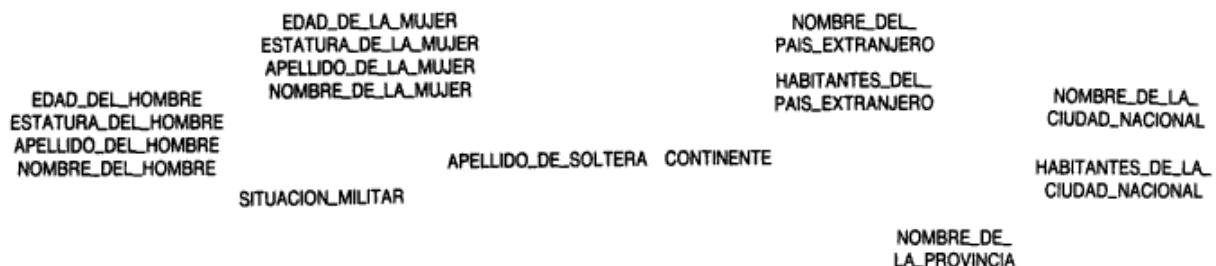


Figura 3.12. La estrategia descendente.

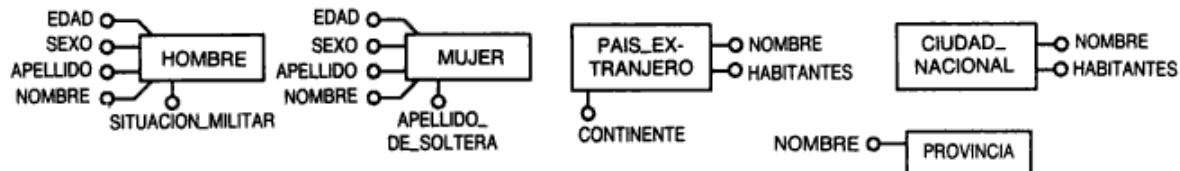
fragmentos de un esquema correspondientes a agrupamientos de conceptos y luego conectarlos.

La ventaja de la estrategia ascendente es su sencillez: al atacar fragmentos del problema completo cada vez, se pueden producir rápidamente versiones preliminares de productos intermedios del diseño. La mayor desventaja del enfoque ascendente es la necesidad de reestructurar el esquema (se mostró un caso simple de reestructuración en la Fig. 3.8). Cuando se integran esquemas complejos, la determinación de las acciones apropiadas de reestructuración es generalmente difícil y resulta crucial. Una visión estable de los conceptos y de las interrelaciones semánticas entre ellos se obtiene sólo al final del proceso de diseño. También existe la posibilidad de obtener esquemas diferentes como resultado de diseños descendentes y ascendentes completos. Esto se asemeja a lo que ocurre en el diseño ascendente de software.

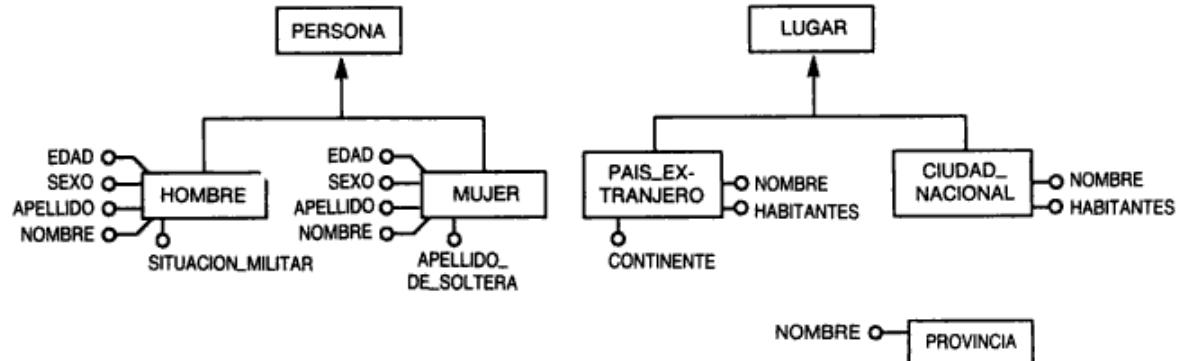
Podemos visualizar las diferencias entre los enfoques descendente y ascendente como sigue: el enfoque descendente permite ver con mucha claridad el bosque pero no los árboles, mientras que el enfoque ascendente permite ver con claridad los árboles pero no el bosque.



(a) Primer esquema



(b) Segundo esquema



(c) Tercer esquema

Figura 3.13. Una sesión de diseño usando la estrategia ascendente.

3.2.3. Estrategia centrífuga

La estrategia centrífuga es un caso especial de estrategia ascendente. Primero se fijan los conceptos más importantes o evidentes, y luego se procede con un movimiento similar al de una mancha de aceite, seleccionando primero los conceptos más cercanos al concepto inicial, y *navegando* después hacia los más distantes. La figura 3.14 representa esta estrategia de diseño, y la figura 3.15 muestra

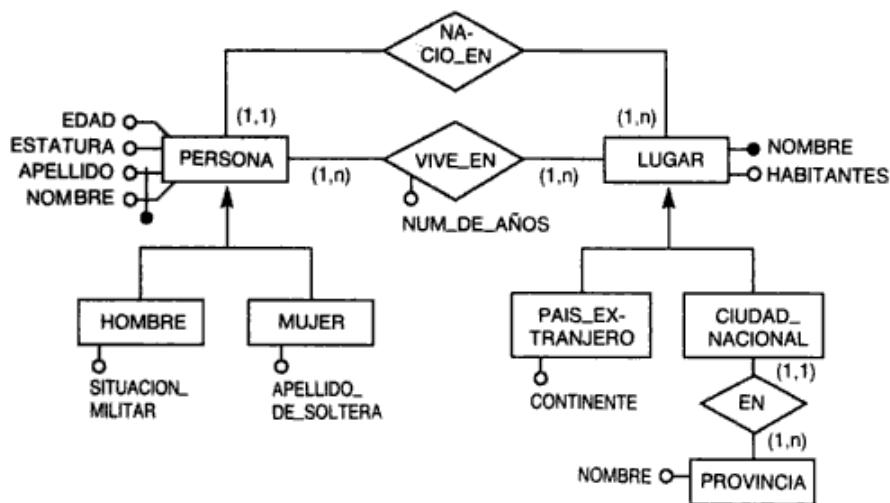
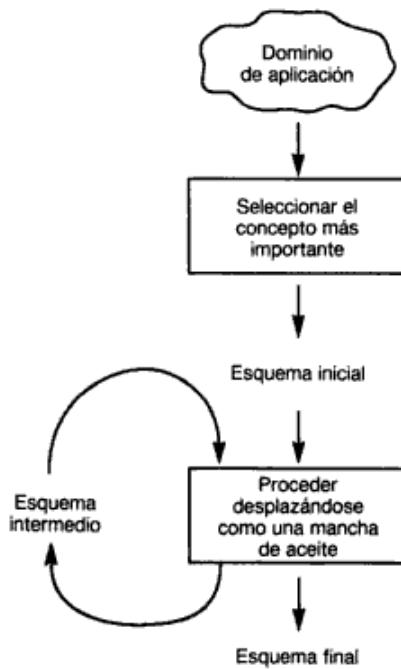
(d) Esquema final (primitiva B₂)Figura 3.13.Bis Una sesión de diseño usando la estrategia ascendente (*continuación*).

Figura 3.14. Estrategia centrífuga.

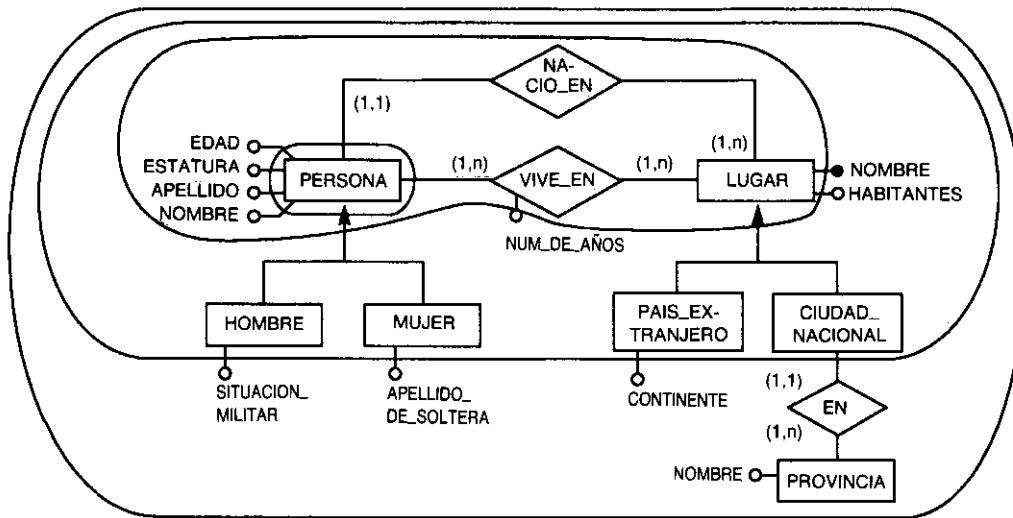


Figura 3.15. Sesión de diseño usando la estrategia centrífuga.

cómo se aplica este enfoque a la base de datos demográfica. PERSONA es el concepto más importante de este ejemplo; si nos desplazamos de la entidad PERSONA hacia fuera, primero introducimos las entidades directamente relacionadas con las personas mediante interrelaciones y generalizaciones; de esta forma, descubrimos la entidad LUGAR, la interrelación VIVE_EN, las entidades HOMBRE y MUJER, y la generalización que les relaciona con PERSONA.

En la figura 3.15, las distintas áreas delineadas representan las diversas capas introducidas progresivamente en el esquema. Ahora se puede iterar la navegación y buscar conceptos cercanos a las entidades recién descubiertas. Esto conduce a representar: 1) la nueva interrelación NACIO_EN, entre PERSONA y LUGAR; 2) las nuevas entidades PAIS_EXTRANJERO y CIUDAD_NACIONAL, que se relacionan con LUGAR en una generalización; y 3) dos nuevos atributos de las entidades HOMBRE y MUJER. Finalmente, se puede introducir la entidad PROVINCIA y la interrelación que la une a CIUDAD_NACIONAL. Luego, se completa el esquema con las cardinalidades y los identificadores.

En la estrategia centrífuga, el orden de los refinamientos es disciplinado, igual que en el enfoque descendente. Sin embargo, los niveles de abstracción de los conceptos introducidos en versiones sucesivas del esquema son similares; de este modo, la ventaja de proceder por niveles de abstracción se pierde.

3.2.4. Estrategia mixta

La estrategia mixta aprovecha tanto la estrategia descendente como la ascendente, al permitir particiones *controladas* de los requerimientos. La idea prin-

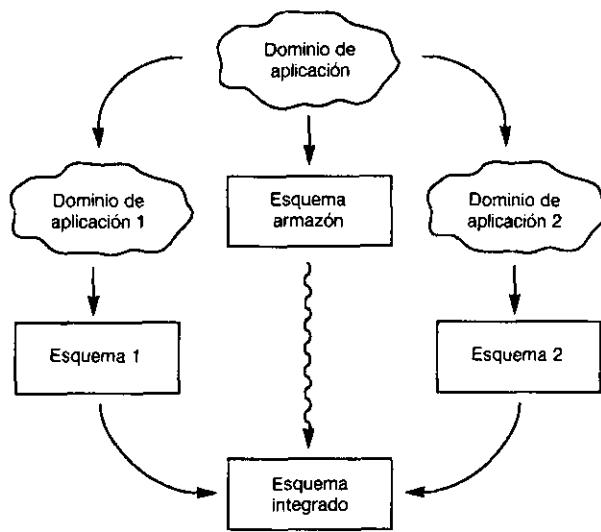


Figura 3.16. La estrategia mixta.

cipal de este enfoque se muestra en la figura 3.16: cuando el dominio de aplicación es muy complejo, el diseñador divide los requerimientos en subconjuntos, que más tarde se consideran por separado. Al mismo tiempo, el diseñador produce un *esquema armazón* que actúa como marco para los conceptos más importantes del dominio de aplicación y fija los nexos entre las particiones. El tiempo adicional requerido por este paso es recompensado, ya que la existencia del esquema armazón permite una integración ascendente más fácil de los diferentes esquemas producidos.

La figura 3.17 muestra el diseño de la base de datos demográfica con el enfoque mixto. Los requerimientos se consideran divididos en dos partes, referentes a personas y lugares, respectivamente. Antes de empezar el diseño por separado de los dos esquemas, se puede identificar con facilidad los conceptos de persona y lugar como aspirantes a entidades del esquema armazón, el cual se puede completar con la interrelación RELACIONADO_CON (Fig. 3.17a). Luego, se continúa con la producción de los dos esquemas para PERSONA y LUGAR (Figs. 3.17b y 3.17c), y se conectan al esquema armazón (Fig. 3.17d). Para terminar, se refina la interrelación RELACIONADO_CON para dar las interrelaciones NACION_EN y VIVE_EN, y se añaden las cardinalidades y los identificadores.

3.2.5. Comparación de estrategias

Este apartado ha presentado cuatro estrategias alternativas y complementarias para el diseño progresivo de esquemas conceptuales. La tabla 3.1 resume y

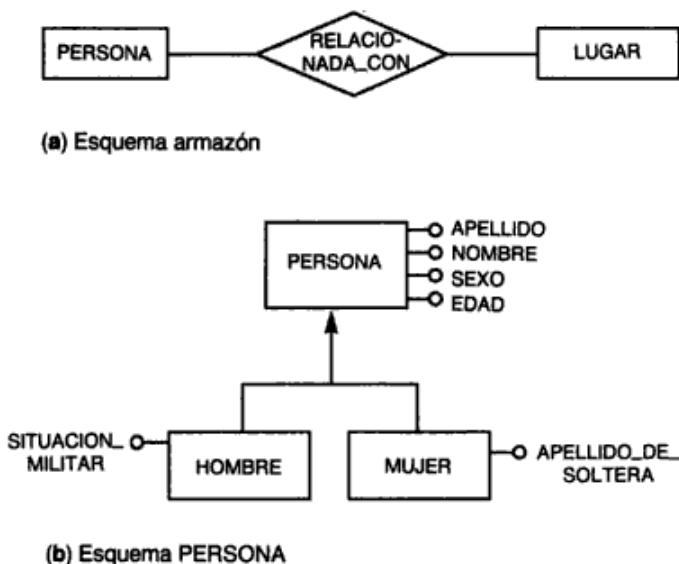


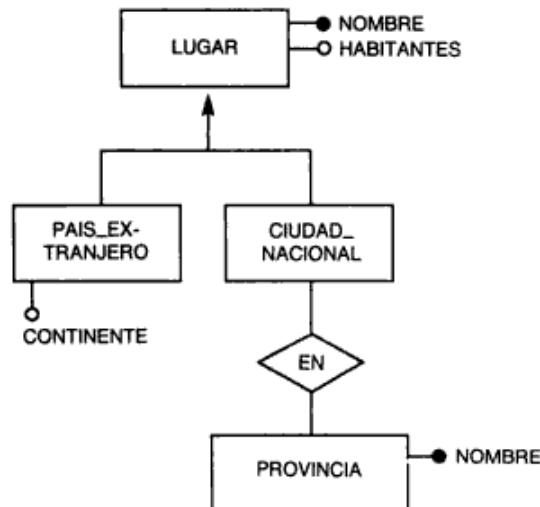
Figura 3.17. Sesión de diseño usando la estrategia mixta.

compara las ventajas y desventajas de las cuatro estrategias. A continuación se plantean algunos problemas relacionados.

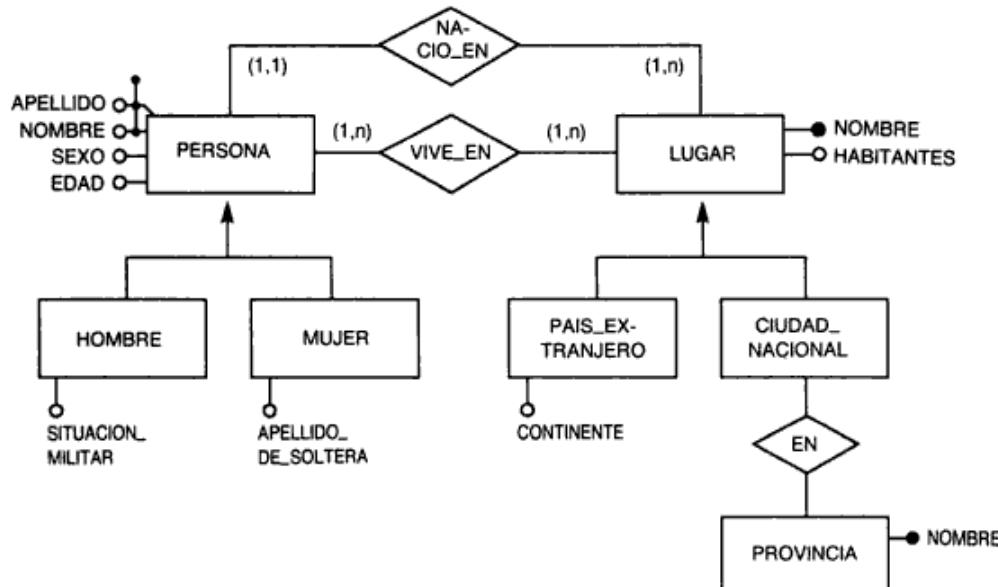
La primera pregunta que surge es la siguiente: A partir de ciertos requerimientos dados ¿conducen siempre las cuatro estrategias al mismo esquema final? La respuesta es claramente negativa, ya que cada estrategia sigue una filosofía de diseño específica. De hecho, incluso adoptando la misma estrategia es posible modelar los mismos requerimientos de maneras muy diferentes. Como consecuencia, es importante efectuar periódicamente una revisión de la calidad del esquema. Este proceso se estudia en el capítulo 6.

Una segunda pregunta se refiere a la aplicabilidad práctica de las estrategias en diferentes organizaciones y entornos. Cada sesión de diseño debe emplear la estrategia más conveniente con respecto al entorno específico. Por ejemplo, una estrategia descendente puede ser la más conveniente en el caso de organizaciones altamente estructuradas, donde los ejecutivos de alto nivel tienen una visión completa del dominio de aplicación en un alto nivel de abstracción. Por otro lado, una estrategia ascendente puede ser conveniente en organizaciones informales y de bajo nivel de estructuración, donde es más fácil modelar la información en detalle y luego integrarla que crear una abstracción inicial de toda la base de datos y luego refinirla.

Se concluye que la estrategia descendente debe usarse siempre que sea posible. De otra forma, se debe adoptar una mezcla de estrategias: las que resulten más naturales en el contexto específico de la aplicación. Un diseñador experimentado puede, en realidad, usar una mezcla de las estrategias para cada uno de los procesos de diseño. Por ejemplo, el diseñador puede elegir proceder de



(c) Esquema LUGAR



(d) Esquema integrado

Figura 3.17.Bis Sesión de diseño usando la estrategia mixta (*continuación*).

forma descendente, porque de esta forma todo el proceso de diseño es bastante estructurado y disciplinado. Sin embargo, en un alto nivel de refinamiento, el diseñador puede olvidar algunos conceptos; éstos se añadirían, por lo regular, aplicando primitivas ascendentes. En otras palabras, una sesión de diseño de ese

Tabla 3.1. Comparación de las estrategias para el diseño de esquemas.

<i>Estrategia</i>	<i>Descripción</i>	<i>Ventajas</i>	<i>Desventajas</i>
Descendente	Los conceptos se refinan progresivamente	No hay efectos secundarios indeseables	Requiere un diseñador hábil con alta capacidad de abstracción desde el comienzo
Ascendente	Los conceptos se crean a partir de componentes elementales Decisiones locales de diseño sencillas	Necesidad de una reestructuración después de aplicar cada primitiva ascendente	
Centrífuga	Ninguna carga sobre el diseñador inicial		
	Los conceptos se construyen con un enfoque «de mancha de aceite»	Facilidad para descubrir nuevos conceptos cercanos a los anteriores	La visión global del dominio de aplicación se tiene sólo al final
Mixta	Partición descendente de los requerimientos; integración ascendente usando el esquema armazón	Ninguna carga sobre el diseñador inicial Enfoque de «divide y vencerás»	Requiere decisiones cruciales sobre el esquema armazón al inicio del proceso de diseño

estilo se conduciría principalmente de forma descendente, con unas pocas aplicaciones excepcionales de primitivas ascendentes.

Recuérdese lo dicho en el apartado 3.1.3, donde se dejó establecido que sólo esquemas producibles en forma descendente pueden definirse mediante la aplicación exclusiva de primitivas descendentes. Sin embargo, la mayoría de los esquemas prácticos de bases de datos (los que se usan en las aplicaciones reales), son producibles en forma descendente. Así pues, una estrategia descendente pura es aplicable a la mayoría de los problemas de la realidad.

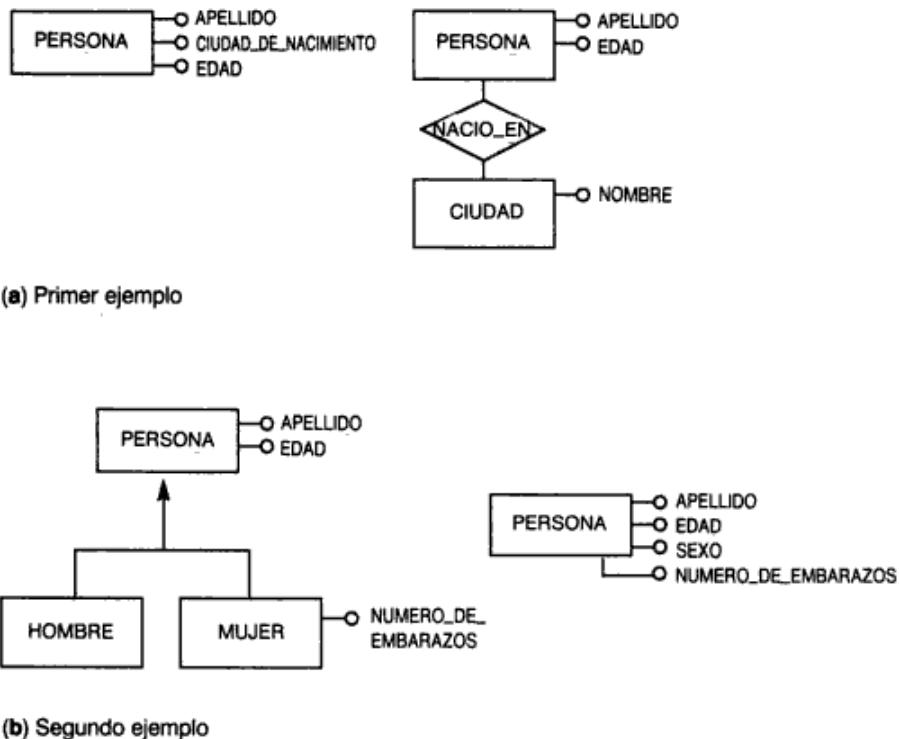


Figura 3.18. Esquemas diferentes para representar la misma realidad.

3.3. Criterios de elección entre conceptos

Modelar objetos de la realidad es un problema crucial, normalmente abierto a varias soluciones alternativas; se sabe que la misma realidad se puede modelar de formas diversas. Por ejemplo, la figura 3.18 muestra dos ejemplos diferentes de esquemas que representan la misma realidad. En este corto apartado, se consideran algunas de las elecciones típicas que afronta un diseñador y se proporcionan algunos criterios sencillos para realizar esas elecciones.

Entidad o atributo simple. Este problema implica elegir si un objeto de la realidad debe ser modelado como una entidad o como un atributo. Se debe elegir una entidad cuando se entiende que varias características (atributos, interrelaciones, generalizaciones, subconjuntos) se pueden asociar con el objeto a modelar, sea ahora o más adelante en el proceso de diseño. Se debe elegir un atributo cuando el objeto posea una estructura atómica simple y no le sea aplicable ninguna propiedad de interés. Por ejemplo, el concepto «color» es por lo regular

un atributo (por ejemplo, para la entidad COCHE); sin embargo, si la aplicación se refiere a la construcción de muebles, y en particular al proceso de coloreado, entonces COLOR bien puede convertirse en una entidad, con los atributos NOMBRE, CODIGO_DE_COLOR, NUMERO_NECESSARIO_DE_PINTURAS, ANTICORROSIVO y así sucesivamente.

Generalización o atributo. Debe usarse una generalización cuando se espera que (ahora o más adelante en el proceso de diseño) alguna característica se asocie con las entidades de más bajo nivel (como un atributo; por ejemplo, NUMERO_DE_EMBARAZOS en el ejemplo de la figura 3.18, o una interrelación con otras entidades); de lo contrario, se elige un atributo. Por ejemplo, una generalización de PERSONA basada en COLOR_DEL_CABELLO no suele ser útil, porque pocas veces se asocian rasgos específicos a los rubios o a los canosos; sin embargo, esta generalización puede servir para una base de datos de estilistas, donde el tratamiento del cabello depende de los colores de éste.

Atributo compuesto o conjunto de atributos simples. Se elige un atributo compuesto cuando resulta natural asignarle un nombre; se elige un conjunto de atributos simples cuando ellos representan propiedades independientes. Por ejemplo, DIRECCION es una buena abstracción de los atributos CALLE, CIUDAD, PROVINCIA, CODIGO_POSTAL.

En la exposición precedente, cuando se dice «ahora o más adelante en el proceso de diseño», se quiere decir que, al realizar una actividad de modelado, se debe, de alguna forma, considerar no sólo la decisión específica que se tiene entre manos, sino también aquello que es inminente o que es probable que ocurra, para así evitar la modificación frecuente de las decisiones tomadas.

3.4. Entradas, salidas y actividades del diseño conceptual

Las primitivas y las estrategias son los pilares para el desarrollo de las metodologías de diseño conceptual; este apartado considera el proceso de diseño de forma global. Inicialmente, se considera una metodología como una caja negra y se examinan sus entradas y salidas, luego se mira dentro de la caja y se presentan las principales fases de diseño que deben pertenecer a una metodología para el diseño conceptual de bases de datos. El próximo capítulo ofrece metodologías específicas, adaptadas a tipos particulares de requerimientos iniciales. La figura 3.19 muestra las entradas y salidas típicas de una metodología para el diseño conceptual.

3.4.1. Entradas

Las entradas al proceso de diseño son los **requerimientos**, es decir, las descripciones de la aplicación de la base de datos: entrevistas, documentos, formularios

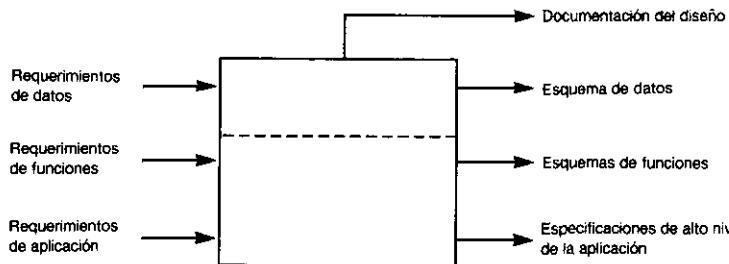


Figura 3.19. Entradas y salidas de una metodología para el diseño conceptual.

y cualquier otra fuente que ofrezca información útil para el proceso de diseño. Las entradas se pueden clasificar de varias formas. Una clasificación inicial se refiere a los tipos de información que ellas describen:

1. **Los requerimientos de datos** describen la estructura de los datos que se debe almacenar dentro de la base de datos (por ejemplo, los empleados tienen un nombre, un salario y un número de seguro social).
2. **Los requerimientos de funciones** describen la estructura dinámica del sistema de información, identificando varias funciones o actividades dentro del sistema (por ejemplo, procesamiento de pedidos, administración de embarques) y los flujos de información entre ellos (por ejemplo, pedidos, notas de embarque). Estos términos se definen ampliamente en el capítulo 8.
3. **Los requerimientos de aplicación** describen operaciones sobre los datos (por ejemplo, inserciones, actualizaciones, consultas: «insertar un empleado nuevo», «modificar el salario de un empleado», «recuperar los salarios de todos los empleados»).

Una segunda clasificación de las entradas se da en términos de la representación lingüística usada en su descripción. Esta clasificación se aplica a los requerimientos de datos, pero una clasificación similar se puede hacer para los flujos de datos y también para los requerimientos de aplicación. Los requerimientos se expresan en diversos «lenguajes»:

1. *Lenguaje natural*, usado en entrevistas y diversos tipos de documentos.
2. *Formularios*, que son hojas usadas para recabar datos e intercambiar información entre los usuarios.
3. *Formatos de registros, divisiones de datos en Cobol*, etc., los cuales describen la estructura de los datos en los sistemas de archivos tradicionales. Se les debe tener en cuenta (junto con los *formatos de pantallas*, por ejemplo, las pantallas mostradas a los usuarios) cuando se desarrolla una base de datos a partir de un sistema de archivos tradicional.
4. *Esquemas de datos*, expresados en un lenguaje de descripción de datos, que

describen la estructura de la información en bases de datos existentes; se los debe tener en cuenta si se quiere cambiar el DBMS, modificar una aplicación de bases de datos existente o fusionar varias aplicaciones de bases de datos en una sola.

Estos ejemplos no agotan la variedad de entradas que se puede usar: por ejemplo, las tablas estadísticas son requerimientos típicos de entrada para bases de datos estadísticas, los mapas se usan como requerimientos de entrada en bases de datos geográficas, etcétera.

3.4.2. Salidas

Las salidas producidas por una metodología de diseño conceptual incluyen: 1) el *esquema conceptual de datos* o *esquema conceptual*, que describe todos los datos presentes en los requerimientos; 2) los *esquemas de funciones*, que describen funciones y actividades del sistema de información y los flujos de información entre ellas; 3) las *especificaciones de alto nivel de la aplicación*, que describen las operaciones efectuadas con la base de datos; 4) otros *documentos de diseño*, que ofrecen información adicional sobre cada una de las salidas anteriores. Son útiles como documentación de la actividad de diseño y para el mantenimiento de la base de datos, cuando los requerimientos cambiantes hagan necesario reestructurar el esquema conceptual.

Obsérvese, en la figura 3.19, que el diseño de datos utiliza como entrada sólo los requerimientos de datos, y produce como salida sólo el esquema conceptual de datos, con documentos de diseño relacionados. Sin embargo, el diseño conjunto funcional y de datos, descrito en la segunda parte de este libro, utiliza todas las entradas y produce todas las salidas.

3.4.3. Actividades

Ahora podemos ver dentro de la caja, al describir las actividades típicas del diseño conceptual.

Análisis de requerimientos. Durante el análisis de requerimientos, el diseñador debe estudiar minuciosamente los requerimientos y trabajar despacio y con cuidado para empezar a producir un esquema conceptual. El objetivo fundamental de esta fase es dar a los requerimientos una estructura que facilite las posteriores actividades de modelado (este problema se trata en el capítulo 4). El diseñador debe eliminar las ambigüedades en los requerimientos (descripciones imprecisas o incorrectas de la realidad), con la intención de producir descripciones claras como entradas para el diseño conceptual.

Conceptualización inicial. El objetivo de esta actividad es hacer una primera selección de los conceptos que se van a representar en el esquema concep-

tual. Esta actividad es típica de las estrategias descendentes, mixtas, y hasta cierto punto las centrífugas, pero se omite en una estrategia ascendente pura. En este nivel, el diseñador crea un conjunto preliminar de abstracciones, buenas candidatas para ser representadas como entidades, interrelaciones y generalizaciones. El esquema producido es en gran medida incompleto, porque representa sólo algunos aspectos de los requerimientos.

Conceptualización incremental. Esta es la actividad central del diseño conceptual. Usando las estrategias generales, descritas en el apartado 3.2, un esquema preliminar se refina para dar lugar a un esquema conceptual final.

Integración. Esta actividad es típica de las estrategias mixtas y ascendentes. Implica la fusión de varios esquemas y la producción de una nueva representación global de todos ellos. Durante la integración, se determinan los elementos comunes de distintos esquemas y se descubren los *conflictos* (esto es, representaciones diferentes de los mismos conceptos) y las *propiedades interesquemáticas* (es decir, restricciones entre esquemas distintos).

Reestructuración. Como en el diseño de software, a veces es recomendable suspender el proceso de diseño en la etapa conceptual y dedicar algo de tiempo a medir y mejorar la calidad del producto obtenido. Sin embargo, ¿qué es un buen esquema? ¿Cuáles son las cualidades pertinentes en el diseño conceptual? Se responde a estas preguntas en el capítulo 6.

Las actividades anteriores son típicas de cualquier proceso de diseño. La importancia de cada una depende en gran medida de la situación específica de diseño. Por ejemplo, si los requerimientos se expresan en lenguaje natural, con muchas ambigüedades y omisiones, es conveniente evitar un análisis profundo de los requerimientos y proceder a la conceptualización inicial. En cambio, si los requerimientos se expresan mediante el uso de formularios, es conveniente hacer un análisis preciso de su estructura. Esto permite una traducción directa de los requerimientos a un esquema conceptual.

3.5. Resumen

En este capítulo se ha expuesto un marco metodológico para el diseño de bases de datos. En primer lugar, se examinaron, a nivel microscópico, las transformaciones elementales usadas para un refinamiento y enriquecimiento progresivos del esquema conceptual. Luego, se examinaron, a nivel macroscópico, las estrategias que puede utilizar un diseñador para crear un esquema conceptual. No se indicó ninguna preferencia definitiva en la elección de una estrategia de diseño; más bien, se mostraron los pros y contras de cada opción. Asimismo, se estudiaron las características generales de las metodologías de diseño en términos de sus entradas, salidas y principales actividades. El próximo capítulo des-

cribirá metodologías de diseño específicas para clases particulares de requerimientos de entrada.

Ejercicios

3.1. Produzca un esquema que incluya, al menos, lo siguiente:

- a) Cinco entidades, una de ellas con dos identificadores diferentes.
- b) Cinco interrelaciones, con una interrelación ternaria.
- c) Una jerarquía de generalización y un subconjunto.

Al finalizar, analice su trabajo e indique qué primitivas y estrategias ha usado. Escriba una descripción narrativa del esquema.

3.2. Considere el esquema conceptual de la base de datos de fútbol de la figura 2.34. Produzca este esquema usando las siguientes estrategias:

- a) Estrategia descendente.
- b) Estrategia ascendente.
- c) Estrategia centrífuga.
- d) Estrategia mixta.

Para cada estrategia, liste la información proporcionada en cada paso y las primitivas usadas.

3.3. Considere el esquema conceptual de la base de datos de proyectos de investigación de la figura 2.35. Produzca este esquema usando estas estrategias:

- a) Estrategia descendente.
- b) Estrategia ascendente.
- c) Estrategia centrífuga.
- d) Estrategia mixta.

3.4. Considere los siguientes requerimientos referidos a la organización de un curso. Los requerimientos están escritos en un estilo centrífugo: comienzan describiendo los participantes de un curso y continúan con todos los otros aspectos importantes.

- Para cada participante en un curso, almacene el nombre, apellido, fecha de nacimiento y sexo.
- Indique si cada participante está casado y el número de hijos.
- Represente, asimismo, las ciudades de residencia y la ciudades donde nacieron, con las provincias.
- Almacene otra información relativa al curso al que asistieron (número de clases, fecha, temas) y al profesor (o profesores) que les enseñó (nombre, apellido, afiliación).

- Para profesores afiliados a universidades, indique su universidad de afiliación y su campo de especialización.

Cree un esquema usando la estrategia centrífuga. Luego, diseñe de nuevo el esquema, usando las estrategias mixta y descendente.

3.5. Estudie los siguientes requerimientos de datos para una base de datos de reservas y cree un esquema conceptual para este dominio de aplicación, usando estas estrategias:

- a) Estrategia descendente.
- b) Estrategia ascendente.
- c) Estrategia centrífuga.
- d) Estrategia mixta.

Quizá necesite hacer ciertas suposiciones sobre los requerimientos de la aplicación; haga suposiciones razonables conforme avance. La base de datos de reservas almacena información sobre vuelos y reservas de pasajeros. Para cada vuelo, se conocen los aeropuertos, fechas y horas de salida y de llegada. Suponga que los vuelos conectan sólo un aeropuerto de salida y un aeropuerto de llegada, sin paradas intermedias. De cada pasajero se conoce el nombre, sexo y número de teléfono; así como el asiento y si fuma o no. Cada pasajero puede tener múltiples reservas.

3.6. Estudie los siguientes requerimientos de datos para una base de datos de un hospital y produzca un esquema conceptual para este dominio de aplicación, usando estas estrategias:

- a) Estrategia descendente.
- b) Estrategia ascendente.
- c) Estrategia centrífuga.
- d) Estrategia mixta.

Quizá necesite hacer ciertas suposiciones sobre los requerimientos de la aplicación; haga suposiciones razonables conforme avance.

La base de datos del hospital almacena información sobre los pacientes, su admisión y alta de los departamentos del hospital, y sus tratamientos. Para cada paciente, se conoce nombre, dirección, sexo, número de seguro social y código del seguro (si lo tiene). Para cada departamento, se conoce el nombre del departamento, su ubicación, el nombre del médico que lo dirige, el número de camas disponibles y el número de camas ocupadas. Cada paciente se admite y se da de alta en una determinada fecha. A cada paciente se le administran varios tratamientos durante la hospitalización; para cada tratamiento, se almacena su nombre, duración y las posibles reacciones que pueda tener el paciente.

Bibliografía

- R. Barker, CASE Method: Entity Relationship Modelling, Addison-Wesley, 1990*.
- S. Ceri (ed.), *Methodology and Tools for Data Base Design*, North-Holland, 1983.
- P. Flatten, D. McCubrey, P. O'Riordan y K. Burgess, *Foundations of Business Systems*, The Dryden Press, 1989.
- M. Lundberg, «The ISAC Approach to the Specification of Information Systems and Its Application to the Organization of an IFIP Working Conference». En Olle, Sol y Verrijn-Stuart, 1982 (ver sección siguiente).
- I. G. MacDonald y I. R. Palmer, «System Development in a Shared Data Environment». En Olle, Sol y Verrijn-Stuart, 1982 (ver sección siguiente).
- D. Ross y K. Shoman, «Structured Analysis for Requirements Definition», *IEEE Transactions on Software Engineering*, SE-3, n.º 1 (1977).
- H. Tardieu, A. Rochfeld y R. Colletti, *Le Methode Merise: Principes et Outils*, París: Les Editions d'Organization, 1983.
- Estos trabajos describen metodologías completas para el diseño de bases de datos que han sido desarrolladas en años recientes y aplicadas extensivamente en proyectos grandes. Se describen otras metodologías en Olle, Sol y Verrijn-Stuart, 1982 (véase más adelante).
- T. W. Olle, H. G. Sol y A. A. Verrijn-Stuart, *Information Systems Design Methodologies: A Comparative Review*, North-Holland, 1982.
- Este libro describe un experimento interesante: la comparación de metodologías existentes para el diseño de sistemas de información. Las metodologías se aplican al mismo ejemplo y se comparan, usando un marco común de evaluación.
- C. Batini y G. Santucci, «Top-down Design in the Entity-Relationship Model». En P. P. Chen (ed.), *Entity-Relationship Approach to System Analysis and Design*, North-Holland, 1980.
- S. Ceri, G. Pelagatti y G. Bracchi, «Structured Methodology for Designing Static and Dynamic Aspects of Data Base Applications», *Information Systems*, 6 (1981): 31-45.
- Estos dos artículos presentan dos prototipos de metodologías puras ascendente y descendente, respectivamente.
- T. J. Teorey, D. Yang y J. P. Fry, «A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model», *ACM Computing Surveys*, 18, n.º 2 (1986): 197-222.
- Este artículo describe una metodología para el diseño de bases de datos usando el modelo de entidades-interrelaciones. Se ofrecen pautas sencillas para elegir conceptos y traducir el esquema al modelo relacional.
- M. Blaha, W. Premerlani y J. Rumbaugh, «Relational Database Design Using an Object-Oriented Methodology», *Communications of the ACM*, 31, n.º 4: 414-27.
- La metodología presentada en este artículo usa un modelo orientado a objetos para representar la estructura de los datos en el nivel lógico. La metodología se compara con el enfoque seguido por Teorey, Yang y Fry (1986). Los autores sostienen que su enfoque es superior porque usa un modelo más expresivo y estrategias más limpias.
- N. M. Mattos y M. Michels, «Modeling with KRISIS: The Design Process of DB Appli-

* Existe versión en castellano publicada por Addison-Wesley/Díaz de Santos, 1994. (N. del E.)

cations Reviewed». En F. Lochovsky (ed), *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

Este artículo extiende las metodologías tradicionales para el diseño de bases de datos al diseño de bases de conocimientos. Los cuatro pasos de la metodología corresponden a construir una conceptualización del mundo de la aplicación, estructurar ese mundo mediante el modelo de conocimiento, refinar la representación de la base de conocimientos para mejorar la semántica, y validar el contenido de la base de conocimientos.

Diseño de vistas

El objetivo principal del diseño de vistas es crear un esquema conceptual partiendo de una descripción informal de los requerimientos del usuario. Se utiliza el término vista para referirse a la percepción de una base de datos o de los requerimientos de datos de una aplicación, tal como lo ve un usuario o un grupo de usuarios. El diseño de vistas abarca, típicamente, dos actividades distintas: 1) el análisis de los requerimientos, para captar el significado de los objetos de interés en la aplicación, su agrupación en clases, sus propiedades etc.; 2) la representación de estos objetos, clases y propiedades, usando los conceptos del modelo ER.

La primera actividad está especialmente influida por la naturaleza de los requerimientos; éstos pueden incluir descripciones en lenguaje natural, formularios, formatos de registros y esquemas de datos, que claramente representan una realidad dada de diferentes maneras. Con los requerimientos expresados en lenguaje natural, la estructura de la información puede estar oculta en descripciones ambiguas, incompletas o contradictorias. Con representaciones más estructuradas, a veces es más fácil deducir la estructura subyacente de la información y expresarla en términos de los componentes de un esquema ER; sin embargo, es posible que se omita información importante en estas representaciones. En este capítulo, se describe el diseño de vistas para tres tipos distintos de requerimientos: lenguaje natural, formularios y declaraciones de registros.

Las **descripciones en lenguaje natural** se hacen normalmente por escrito; así, se deduce información sobre la estructura de la base de datos a partir de un *análisis textual* de los requerimientos. En el apartado 4.1, se presentan sugerencias prácticas para analizar y eliminar las ambigüedades de las descripciones textuales.

Un **formulario** es cualquier módulo de papel usado para recolectar datos; en el caso de sistemas de información que ya emplean computadoras, se puede usar también descripciones impresas de *pantallas con formato*, es decir, pantallas que se presentan en una terminal para introducir datos en un programa o base de datos ya existente. En el apartado 4.2 se clasifica la información presente en los formularios y luego se ofrece una metodología para crear un esquema mediante la extracción progresiva de información de los formularios.

Las **declaraciones de registros** o **formatos de registros** pertenecen a apli-

1. Análisis de los requerimientos
 - 1.1. Analizar los requerimientos y filtrar las ambigüedades
 - 1.2. Dividir los enunciados en conjuntos homogéneos
2. Diseño inicial
 - 2.1. Construir un esquema armazón global
3. Diseño de esquemas: para cada concepto del esquema armazón, aplicar
 - 3.1. Primitivas descendentes
 - 3.2. Primitivas ascendentes
 - 3.3. Primitivas centrífugashasta que todos los requerimientos estén expresados en el esquema

Figura 4.1. Diseño de vistas a partir de requerimientos en lenguaje natural.

ciones ya existentes, escritas en lenguajes de programación convencionales. Es importante tener en cuenta este tipo de datos de entrada, porque en muchos casos el sistema de bases de datos incorpora una organización de archivos construida con un lenguaje de programación convencional. En el apartado 4.3, se centra la atención en Cobol, el lenguaje de programación más usado para aplicaciones convencionales de procesamiento de datos.

El análisis de los requerimientos está fuertemente influido por la naturaleza de los requerimientos. Los pasos subsecuentes utilizan las primitivas y estrategias de diseño generales, descritas en los apartados 3.1 y 3.2.

4.1. Diseño de vistas a partir de requerimientos expresados en lenguaje natural

Una metodología para el diseño de vistas a partir del lenguaje natural incluye el análisis de requerimientos, el diseño inicial y el diseño de esquemas, como muestra la figura 4.1. Durante el análisis de los requerimientos, el texto que describe los requerimientos se analiza cuidadosamente para descubrir ambigüedades y para entender en detalle el significado de los términos. Luego se dividen los enunciados en conjuntos homogéneos, de modo que cada conjunto corresponda a un concepto específico. Durante el diseño inicial, estos grupos de enunciados son la base para construir el esquema armazón, que expresa los conceptos e interrelaciones más importantes. Luego, el esquema se refina por medio de primitivas descendentes y ascendentes, hasta que todos los requerimientos estén representados en términos de conceptos de ER.

4.1.1. Análisis de los requerimientos

Sabemos, por experiencia, que el lenguaje natural es ambiguo y que los malentendidos son muy comunes. En el caso de requerimientos escritos en lenguaje

1 En una base de datos de una universidad, se representan datos
2 sobre estudiantes y profesores. Para los estudiantes, se
3 representa el apellido, edad, sexo, ciudad y provincia de
4 nacimiento, ciudad y provincia de residencia de sus
5 familias, lugares y provincias donde vivieron antes
6 (con el lapso que vivieron en cada uno), cursos que han
7 aprobado, con nombre, código, profesor,
8 nota y fecha. Asimismo, se representan los cursos
9 a los que asisten en la actualidad y, para cada uno, día, sitios
10 y horas de impartición de las clases (cada curso
11 se imparte a lo sumo una vez en un día). Para estudiantes graduados,
12 se representa el nombre del consejero
13 y el número total de créditos en el último año.
14 Para estudiantes de doctorado, se representa el título y área
15 de investigación de su tesis. Para los maestros, se
16 representa el apellido, edad, lugar y provincia de nacimiento,
17 nombre del departamento al que pertenecen, número de teléfono,
18 título, situación y temas de investigación.

Figura 4.2. Requerimientos para la base de datos de una universidad.

natural, es conveniente realizar un análisis profundo del texto. Este análisis es aún más necesario cuando los requerimientos se transmiten oralmente, mediante entrevistas o conversaciones informales. Sólo cuando los requerimientos han sido establecidos firmemente, se puede continuar con seguridad. Los ejemplos de este apartado se basan en los requerimientos escritos presentados en la figura 4.2.

Si analizamos en detalle los enunciados de la figura 4.2, encontraremos varias inexactitudes y ambigüedades. ¿Cómo se puede proceder a descubrirlas y filtrarlas? Las siete reglas empíricas que siguen son de utilidad.

1. Elegir el nivel apropiado de abstracción para los términos. Los términos abstractos se usan con frecuencia en enunciados de la vida real, en casos en que los términos específicos serían más apropiados para clarificar la situación. Las categorías generales son comunes en el lenguaje natural porque producen una comunicación rápida y eficaz en la que, comúnmente, la ambigüedad se resuelve por el contexto. Sin embargo, en el diseño conceptual se debe utilizar términos en un nivel correcto de abstracción, especialmente si el diseñador no es un experto en el dominio de la aplicación. En nuestro ejemplo, aparecen los siguientes términos abstractos: lugares, lapso y situación; los términos apropiados correspondientes son: ciudades, número de años, estado civil.

2. Evitar el uso de casos en lugar de conceptos generales. Esta regla evita la fuente opuesta de ambigüedades; los usuarios de los sistemas de información adoptan, a veces, términos más específicos que lo necesario. Por ejemplo, en una empresa de electrónica, un encargado puede decir: «necesito conocer, a diario, la cantidad en existencia de chips». El término *chips* no describe un con-

cepto, sino más bien un caso del concepto correcto, esto es, componentes. Por tanto, el término preferido tendría que ser *componentes*.

3. Evitar las expresiones vagas o indirectas. En el lenguaje natural se usan con frecuencia la repetición deliberada y las expresiones indirectas. Se puede decir: «mira la persona sentada en la taquilla», en vez de: «mira el taquillero». La segunda oración indica una clase específica de entidades (taquillero), mientras que la primera se refiere a la misma clase indicando una interrelación con otra clase de entidades (persona). Así pues, la segunda oración permite una clasificación más clara de los conceptos. Al usar rodeos se incurre en el riesgo de expresar el significado de los conceptos en términos de referencias implícitas a otros conceptos, en lugar de referencias explícitas a los conceptos mismos.

4. Elegir un estilo estandarizado de enunciado. En la libre conversación se usan muchos estilos sintácticos para lograr una comunicación más eficaz. Esta variedad de estilos debe evitarse en los textos que definen los requerimientos; el uso de categorías sintácticas simples permite un modelado directo (y único) de los requerimientos. Idealmente, se deberían producir enunciados que respondan a algún estilo estándar; por ejemplo, las descripciones de los datos deberían ser de la forma *<sujeto> <verbo> <especificación>*. Los enunciados que describen operaciones deberían utilizar, tanto como sea posible, estructuras sintácticas no ambiguas, similares a las de los lenguajes de programación, como *<si> <condición> <entonces> <acción> <if no> <acción>* o *<cuando> <condición> <hacer> <acción>*. La aplicación completa de esta regla no es siempre posible o conveniente; el diseñador debe seleccionar un estilo apropiado como un término medio entre la estandarización y la expresividad.

5. Verificar los sinónimos y homónimos. Los requerimientos suelen resultar de las contribuciones de varios usuarios. Distintas personas pueden dar el mismo significado a diferentes palabras (*sinónimos*) o diferente significado a las mismas palabras (*homónimos*). En general, el riesgo de los homónimos es mayor cuando el vocabulario de términos es pequeño, mientras que el riesgo de los sinónimos es mayor cuando el vocabulario de términos es rico. Más aún, si dos usuarios distintos adoptan vocabularios en diferentes niveles de abstracción, incurren en el riesgo de los sinónimos. En el ejemplo, los tres términos diferentes: *maestro*, *profesor* y *consejero* se refieren al mismo concepto (son sinónimos). *Lugares* se usa dos veces, con diferentes significados (homónimo).

6. Hacer explícitas las referencias entre términos. Algunas ambigüedades surgen al no especificar las referencias entre los términos. En el ejemplo, no está claro si el *número de teléfono* es una propiedad de los profesores o de los departamentos. Nótese que los conceptos referidos pueden estar explicitamente mencionados en los requerimientos (*profesores* y *departamentos*) u omitidos por completo (esto es cierto para *día* que se puede interpretar como *día de la se-*

Línea	Término	Nuevo término	Razones para la corrección
5	Lugares	Ciudades	<i>Lugar</i> es una palabra genérica
6	Lapso	Núm. de años	<i>Lapso</i> es una palabra genérica
9	Actualidad	Año actual	<i>Actualidad</i> es ambigua
9	Dia	Día de la semana	Más específico
9	Sitios	Aulas	Homónimo de <i>lugares</i> en la línea 5
10	Clases	Cursos	Sinónimo de <i>cursos</i> en la línea 8
15	Maestro	Profesor	Sinónimo de <i>profesor</i> en la línea 2
16	Lugar	Ciudad	Lo mismo que en la línea 5
17	Teléfono	Teléfono del departamento	Más específico
18	Situación	Estado civil	<i>Situación</i> es ambiguo
18	Tema	Área de investigación	Sinónimo de <i>área de investigación</i> en la línea 15

Figura 4.3. Términos ambiguos en los requerimientos con posibles correcciones.

mana o día del mes; los términos semana y mes no aparecen en los requerimientos).

7. Utilizar un glosario. La creación de un **glosario de términos** es una buena forma (aunque demanda bastante tiempo) de entender el significado de los términos y eliminar las ambigüedades de los requerimientos. Después de crear un glosario completo, sólo se deberían utilizar los términos del glosario en las descripciones de los requerimientos. El glosario debe incluir, para cada término: 1) su nombre; 2) una definición corta (5 a 20 palabras) que sea aceptable para todos los usuarios del término; 3) posibles sinónimos, es decir, términos que tengan igual significado para los usuarios (los sinónimos expresan el área de equivalencia del término); y 4) posibles *palabras clave*, es decir, palabras lógicamente cercanas al término (las palabras clave expresan el área de influencia del término).

La aplicación de estas reglas usualmente produce requerimientos más estructurados que al inicio de la actividad de diseño. La figura 4.3 muestra todas las fuentes de ambigüedad y sus correcciones. La figura 4.4 muestra los requerimientos escritos de nuevo.

Llegados a este punto, se analiza el texto y se descompone en conjuntos de enunciados, tales que cada conjunto de enunciados se refiera al mismo concepto; este proceso se muestra en la figura 4.5. Esta actividad produce modificaciones locales del texto o el desplazamiento de fragmentos del mismo, y ayuda a estructurar los requerimientos. Si los enunciados relativos al mismo concepto están agrupados, es más fácil tener en cuenta todos los detalles acerca de ese concepto durante el diseño.

En una base de datos de una universidad se representan datos sobre estudiantes y profesores. Para los estudiantes se representa el apellido, edad, sexo, ciudad y provincia de nacimiento, ciudad y provincia de residencia de sus familias, ciudades y provincias donde han vivido antes (con el número de años que vivieron en cada una), cursos que han aprobado, con nombre, código, profesor, nota y fecha. Asimismo, se representan los cursos a los que asisten en el año actual y, para cada uno, día de la semana, aulas y horas de impartición de los cursos (cada curso se imparte a lo sumo una vez en un día). Para estudiantes graduados se representa el nombre del consejero doctorado, se representa el título y área de investigación de sus tesis. Para profesores se representa el apellido, edad, ciudad y provincia de nacimiento, nombre del departamento al que pertenecen, número de teléfono del departamento, título, estado civil y área de investigación.

Figura 4.4. Los requerimientos, después de filtrar las ambigüedades.

4.1.2. Diseño inicial

El objetivo del diseño inicial es la creación de un esquema armazón. Los conceptos que aparecen en el esquema armazón son los conceptos más *evidentes* mencionados en los requerimientos. Primero se considera el agrupamiento de los enunciados determinados durante el análisis de requerimientos: los conceptos mencionados en cada grupo serán buenos candidatos para transformarse en

En una base de datos de una universidad se representan datos sobre estudiantes y profesores.

Enunciados generales

Para los estudiantes, se representa el apellido, edad, sexo, ciudad y provincia de nacimiento, ciudad y provincia de residencia de sus familias, ciudades y provincias donde han vivido antes (con el número de años que vivieron en cada una), cursos que han aprobado, con nombre, código, profesor, nota y fecha.

Enunciados sobre estudiantes

Asimismo se representan los cursos a los que asisten en el año actual y, para cada uno, día de la semana, aulas y horas de impartición de los cursos (cada curso se imparte a lo sumo una vez en un día).

Enunciados sobre cursos

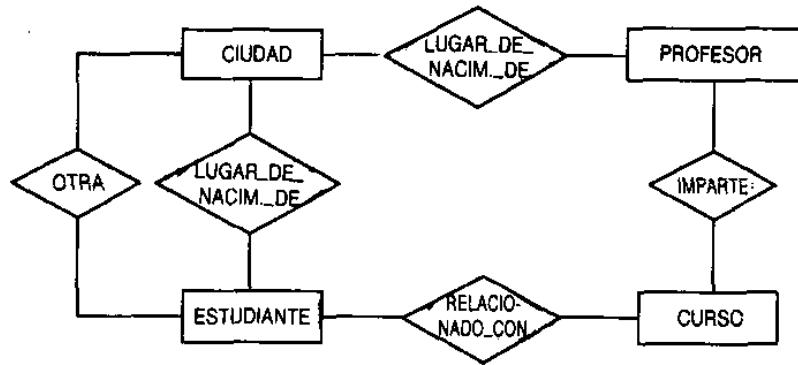
Para estudiantes graduados se representa el nombre del consejero en el último año. Para estudiantes de doctorado se representa el título y área de investigación de sus tesis.

Enunciados sobre tipos específicos de estudiantes

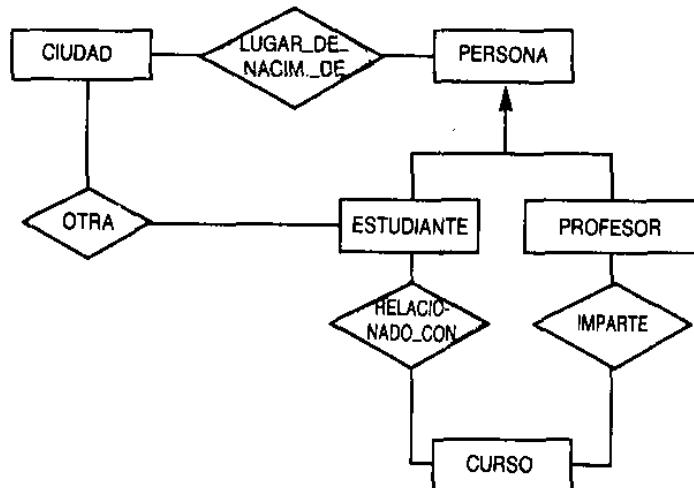
Para profesores se representa el apellido, edad, ciudad y provincia de nacimiento, nombre del departamento al que pertenecen, número de teléfono del departamento, título, estado civil y área de investigación.

Enunciados sobre profesores

Figura 4.5. Partición de enunciados en grupos homogéneos.



(a) Primer esquema armazón



(b) Esquema armazón refinado

Figura 4.6. Desarrollo del esquema armazón.

entidades del esquema armazón. En el ejemplo, son: ESTUDIANTE, PROFESOR y CURSO. Se añade CIUDAD, una entidad fácilmente reconocible.

Una vez que se elige un grupo inicial de entidades, se le puede superponer una red inicial de interrelaciones, correspondientes a enlaces lógicos entre grupos de enunciados. Así, la interrelación LUGAR_DE_NACIMIENTO_DE conecta CIUDAD con ESTUDIANTE y PROFESOR, IMPARTE une PROFESOR y CURSO, RELACIONADO_CON une CURSO y ESTUDIANTE, y OTRA une CIUDAD y ESTUDIANTE. Las dos últimas interrelaciones son, intencionalmente, imprecisas y se refinrarán más adelante. El esquema correspondiente se muestra en la figura 4.6a.

Ya se tiene un primer esquema armazón; antes de continuar con el diseño, es conveniente revisar el esquema armazón y posiblemente realizar alguna reestructuración. Si se observa las interrelaciones LUGAR_DE_NACIMIENTO_DE entre

los pares de entidades (ESTUDIANTE, CIUDAD) y (PROFESOR, CIUDAD), se descubre una semejanza entre PROFESOR y ESTUDIANTE; esta semejanza se confirma si se observa el resto de los requerimientos. Por tanto, modificamos el esquema introduciendo la nueva entidad PERSONA y fusionando las dos interrelaciones LUGAR_DE_NACIMIENTO en una interrelación nueva entre CIUDAD y PERSONA. La introducción de la nueva entidad PERSONA simplifica las posteriores actividades de diseño, puesto que las propiedades comunes a ESTUDIANTE y PROFESOR ahora se relacionarán con PERSONA.

4.1.3. Diseño de esquemas

Ahora se procede a refinar y extender el esquema a fin de representar todas las características expresadas en los requerimientos. El análisis se concentra en cada concepto del esquema armazón, verificando si se puede refinar con el uso de las reglas de refinamiento analizadas en el capítulo 3. En el ejemplo, se usa lo siguiente:

1. Refinamientos descendentes.

- a) La entidad ESTUDIANTE se puede refinar en términos de dos subconjuntos: ESTUDIANTE_GRADUADO y ESTUDIANTE_DE_ESTUDIANTE_DE_DOCENTADO.
- b) La interrelación OTRA, entre las entidades CIUDAD y ESTUDIANTE, se puede refinar en términos de dos interrelaciones: RESIDENCIA y RESIDENCIA_FAMILIAR.

2. Refinamientos ascendentes.

Una vez insertada la entidad ESTUDIANTE_GRADUADO en el esquema, se observan los requerimientos y se nota que existe una relación entre estudiantes graduados y sus profesores asesores. Esta interrelación, llamada ASESORADO_POR, se puede insertar ahora en el esquema.

3. Refinamientos centrífugos.

Una de las propiedades de la entidad PROFESOR es DEPARTAMENTO. Puesto que hay varias de las propiedades asociadas con los departamentos (nombre, dirección y número de teléfono), se puede representar DEPARTAMENTO como una entidad, y el enlace lógico entre PROFESOR y DEPARTAMENTO mediante la interrelación PERTENECE_A. El esquema que resulta de la aplicación de estas primitivas se muestra en la figura 4.7.

Para proceder a hacer los refinamientos finales, ahora se puede enfocar cada concepto del esquema y verificar su compleción. De ese modo, se definen atributos para cada entidad o interrelación y se especifican los identificadores y las correspondencias. Puede verse que los requerimientos textuales no se expresan muy bien con la interrelación RELACIONADO_CON, entre ESTUDIANTE y CURSO. De hecho, la interrelación se debe refinar con la introducción de estas nuevas interrelaciones:

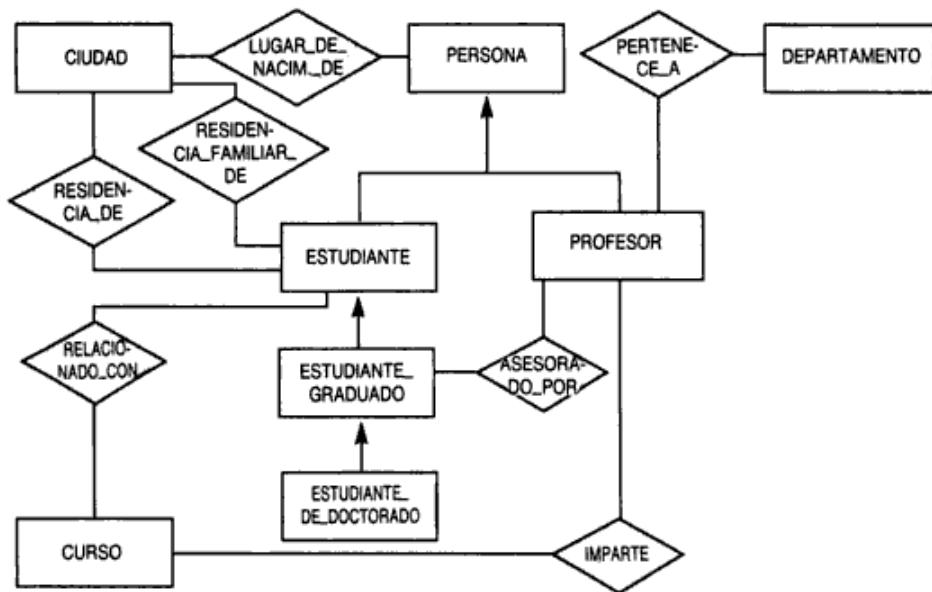


Figura 4.7. Refinamiento del esquema armazón.

1. La interrelación APROBO, la cual representa cursos que el estudiante aprobó, con dos atributos: NOTA y FECHA.
2. La interrelación ASISTE, la cual representa cursos a los que el estudiante asiste actualmente.
3. La interrelación SE_IMPARENTE_EN, entre CURSO y la nueva entidad DIA_DE_LA_SEMANA, que representa el horario semanal de clases a las que asisten los estudiantes durante el año actual, con dos atributos: AULA y HORA.

El esquema se completa añadiéndole algunos otros atributos, cardinalidades de interrelaciones e identificadores. El esquema final se muestra en la figura 4.8.

4.2. Diseño de vistas a partir de formularios

Los *formularios* son documentos estructurados utilizados para intercambiar información dentro de las organizaciones, y en particular para proporcionar información de entrada de datos a los sistemas automatizados. Puesto que los formularios están orientados al usuario, deben ser de fácil comprensión.

Comúnmente, se puede distinguir cuatro partes en un formulario: de certificación, extensiva, intensiva y partes descriptivas. La *parte de certificación* contiene información que certifica la existencia y corrección del formulario, como son los identificadores, fecha de emisión, sellos, marcas y firmas. Por lo regular,

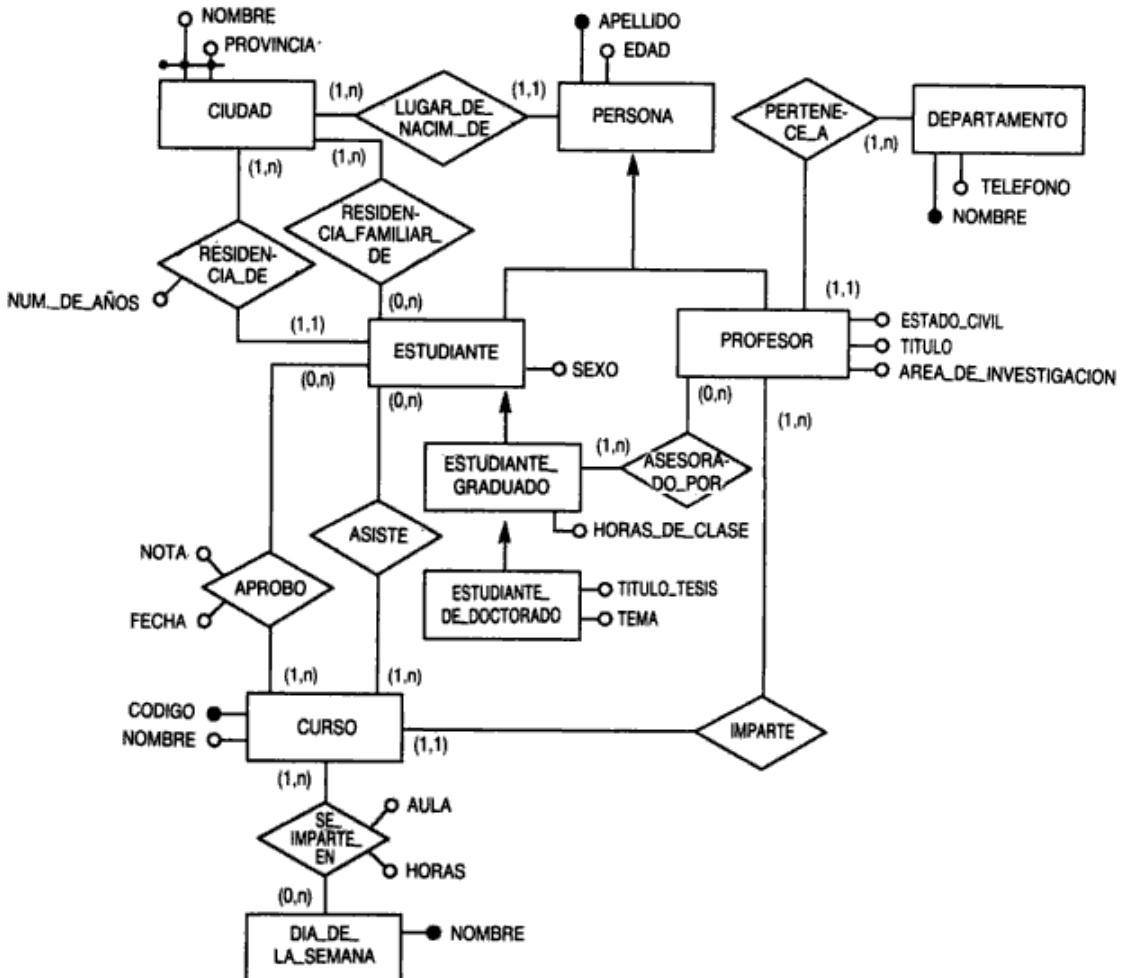


Figura 4.8. Esquema final.

esta parte no comunica información semántica pertinente y no se hará más referencia a ella. La *parte extensiva* es el conjunto de campos que son llenados con valores proporcionados por el usuario cuando se compila el formulario. Esta es la información que una persona incorpora en un formulario preimpreso. La *parte intensiva* es el conjunto de referencias a nombres de campos, explícitas o implícitas, del formulario. Esta es la información preimpresa en los formularios de papel. La *parte descriptiva* contiene instrucciones o reglas que deben seguirse para llenar los campos de la parte extensiva.

La figura 4.9 muestra el Formulario 1040EZ del impuesto sobre la renta para declarantes solteros sin dependientes, emitido por el servicio norteamericano de recaudación interna (U.S. Internal Revenue Service, IRS): las cuatro partes se indican en la figura. Usualmente las partes extensiva e intensiva del formulario

Parte intensiva	
Formulario 1040EZ Declaración de impuestos para declarantes solteros sin dependientes (0) 1989	Departamento del Tesoro - Servicio de Recaudación Interna Use la etiqueta IRS. Si no tiene etiqueta, por favor use letra de imprenta. Escriba su nombre arriba (nombre, inicial, apellido). Dirección (calle y número). (Si tiene apdo. postal, vea al dorso). Ciudad, estado y código postal Parte descriptiva 1 Las instrucciones figuran al dorso. Vea también el formulario 1040A/folleto 1040EX, especialmente la lista de la página 14. Fondo para la campaña electoral presidencial ¿Desea donar 1\$ a este fondo? <small>Nota: Marcar «Sí» no alterará su impuesto ni reducirá su devolución.</small>
Declare sus ingresos Adjunte copia B del formulario W-2 aquí. Nota: Debe marcar Sí o No.	1 Total pagos, salarios y propinas. Esto deberá exponerlo en la casilla 10 del formulario W-2 (Adjunte su forma W-2) 1 2 Ingresos gravables por intereses de 400\$ o menos. Si el total supera los 400\$, no puede usar el formulario 1040EZ. 2 3 Sume las líneas 1 y 2. Estos son sus ingresos brutos ajustados 3 4 ¿Pueden sus padres (o alguna otra persona) incluirlo a U.d. como dependiente en su declaración? <input type="checkbox"/> Sí. Llene la hoja de trabajo al dorso; introduzca la cantidad de la línea E aquí. <input type="checkbox"/> No. Escriba 5,100. Este es el total de su deducción estándar y exención personal. 4 5 Reste la línea 4 de la línea 3. Si la línea 4 es mayor que la 3, escriba 0. Esta es su base gravable . 5 Calcule su impuesto 6 Escriba su impuesto federal sobre la renta retenido de la casilla 9 de su formulario W-2. 6 7 Impuesto. Use la cantidad de la línea 5 para buscar su impuesto en la tabla de las páginas 41-46 del folleto del formulario 1040A/1040EZ. Use la columna marcada soltero de la tabla. Escriba el impuesto en esta línea. 7 Devolución o cantidad que se adeuda Añada aquí el pago de impuestos 8 Si la línea 6 es mayor que la 7, reste la línea 7 de la 6. Esta es su devolución . 8 9 Si la línea 7 es mayor que la 6, reste la línea 6 de la 7. Esta es su cantidad que usted adeuda . Adjunte cheque o giro postal por el total pagadero a Internal Revenue Services. 9 Fírmese su declaración Leída esta declaración, declaro bajo pena de perjurio que, a mi leal saber y entender, la declaración es veraz, correcta y completa. Firma _____ (Guarde una copia de este formulario en su archivo.) X _____
Parte de certificación	
El Acta de Privacidad y el Aviso de Acta de Reducción de Trámites están en la página 3 del folleto . Parte descriptiva 2	
Formulario 1040EZ (1989)	

Figura 4.9. Formulario 1040EZ de declaración de impuestos de Estados Unidos.

1. Análisis de los requerimientos
 - 1.1. Distinga las partes extensiva, intensiva y descriptiva del formulario
 - 1.2. Seleccione áreas y subáreas
2. Diseño inicial
 - 2.1. Construya un esquema armazón global
3. Diseño de esquemas: para cada área
 - 3.1. Construya el esquema del área
 - 3.2. Fusione el esquema de área con el esquema armazón

Figura 4.10. Diseño de vistas a partir de formularios.

se intercalan; las partes descriptivas pueden estar intercaladas o separadas (por ejemplo, las notas al pie). A veces se omite la parte descriptiva, porque las reglas para llenar el formulario son muy evidentes (por ejemplo, el campo APELLIDO_DE_SOLTERA debe ser llenado sólo por mujeres).

Una metodología para el diseño de vistas a partir de formularios se muestra en la figura 4.10. Dentro del análisis de los formularios, se identifican las partes extensiva, intensiva y descriptiva. El formulario se descompone en áreas, es decir, porciones del formulario que son homogéneas en su contenido y describen los mismos conceptos. Luego se desarrolla el esquema armazón seleccionando unos cuantos conceptos para cada área, y el diseño de vistas subsecuente se realiza mediante un análisis área por área.

4.2.1. Análisis de formularios

El primer objetivo del análisis de formularios es entender la estructura y significado del formulario; para este fin, es útil distinguir sus partes extensiva, intensiva y descriptiva. Se obtiene información adicional sobre la estructura de los formularios subdividiéndolos en áreas. Puesto que los formularios se usan para facilitar el intercambio de información, la posición de los campos en los formularios está, por lo regular, muy estudiada, y la información homogénea es contigua. Un *área* es, simplemente, una porción del formulario que trata con elementos de datos estrechamente relacionados entre sí.

Considérese una parte del formulario 1040A para la declaración de impuestos individuales en los Estados Unidos de 1989, mostrado en la figura 4.11. Se distinguen tres áreas: el área 1 se refiere a los datos personales, el área 2 a las exenciones y el área 3 a la evaluación de ingresos. Las áreas pueden dividirse posteriormente en subáreas. En el área 2 se detecta una subárea sobre los dependientes del declarante, y en el área 3 se detecta una subárea sobre la procedencia de las rentas. Como regla general, los diseñadores prefieren usar descomposiciones en áreas que dividan cada formulario en fragmentos de complejidad similar; lo mismo se aplica a la descomposición de áreas en subáreas. Así, las áreas y subáreas de los formularios resultan buenas candidatas para descompo-

ner la actividad de diseño. En la figura 4.11 se asocia un marco con cada área o subárea; el árbol de la figura 4.12 también representa la descomposición del formulario en áreas.

4.2.2. Diseño inicial

En el diseño de un esquema armazón es importante la elección de un grupo de conceptos que estén en un nivel de abstracción apropiado: ni muy general, ni muy detallado. Si los conceptos son demasiado generales, se requiere un gran número de refinamientos para completar el diseño; si los conceptos son demasiado detallados, el esquema armazón no ofrece una visión global de la aplicación. Un buen punto de partida para elegir los conceptos del esquema armazón es organizar las áreas de forma jerárquica, en una estructura de árbol que indique fragmentos de información homogéneos de alto nivel.

La figura 4.13 muestra el esquema armazón e indica, para cada entidad, el área de la que proviene. El esquema armazón incluye las entidades DATOS_PERSONALES, DATOS_DE_EXENCIONES, DATOS_DE_INGRESOS y DETALLE_DE_DATOS_DE_INGRESOS, y las interrelaciones entre ellas.

4.2.3. Diseño de esquemas

Durante el diseño de esquemas, el esquema armazón se transforma y enriquece progresivamente. Tanto la parte intensiva del formulario como la extensiva proporcionan sugerencias útiles sobre cómo proceder en el diseño. En este subapartado se analizan algunas estructuras que se presentan comúnmente en los formularios y se muestra la traducción de estas estructuras a conceptos de ER.

Texto paramétrico. El texto paramétrico es un texto en lenguaje natural con algunos campos vacíos que tienen que llenarse con valores provenientes de dominios adecuados. El texto se completa con indicaciones adicionales sobre los valores a introducir en los campos; tanto el texto como las indicaciones adicionales constituyen la parte intensiva. Cuando los campos se llenan, el texto se vuelve completo y coherente. Un ejemplo de texto paramétrico es el siguiente:

Se certifica que _____, nacido en _____
(Nombre) (Apellido) (Ciudad/Pueblo)
el ____/____/19____, sirvió en el ejército desde el ____/____/19____ hasta el ____/____/19____
como _____
(Oficial/Soldado)

Adviértanse los distintos tipos de subtítulos usados en el texto paramétrico para expresar las propiedades de los datos. En la primera línea del texto, *Nom-*

1989 Régimen 1 (Formulario 1040A)

OMB núm. 1545-0085

Nombre(s) puestos en el formulario 1040A. (No rellene si aparece(n) en la otra cara).

Su número de seguro social

Parte 1

(continued)

Rellene las líneas 13 a 20 solamente si recibió de su patrón subsidios de asistencia a sus dependientes. Asegúrese de completar también las líneas 1 y 2 de la parte I.

- | | |
|--|----|
| 13 Introduzca la suma total de subsidios de asistencia a sus dependientes provistos por su patrón en 1989.
(Esta cantidad debe aparecer por separado en el formulario W-2 etiquetada como «DCB».) NO INCLUYA las cantidades informadas como pagos en la casilla 10 del formulario W-2. | 13 |
| 14 Introduzca la suma de gastos calificados realizados durante 1989 para el cuidado de una persona que califique. (Vea página 24 de las instrucciones.) | 14 |
| 15 Compare las cantidades de las líneas 13 y 14 e introduzca la menor de las dos. | 15 |
| 16 Debe introducir sus ingresos obtenidos . (En la página 34 de las instrucciones está la definición de ingresos obtenidos.) | 16 |
| 17 Si estuvo casado al final de 1989, deberá introducir los ingresos obtenidos por su cónyuge. (Si su cónyuge es estudiante de tiempo completo o incapacitado, consulte en la pág. 34 de las instrucciones la cantidad a introducir.) | 17 |
| 18 • Si estuvo casado al final de 1989, compare las cantidades de las líneas 16 y 17 e introduzca la menor de las dos.
• Si no estuvo casado, introduzca la cantidad de la línea 16. | 18 |
| 19 Beneficios excluidos. Introduzca la menor de las siguientes:
• La cantidad de la línea 15, o
• La cantidad de la línea 18, o
• \$5000 (\$2500 si es casado y rellena una declaración aparte). | 19 |
| 20 Beneficios gravables. Reste la línea 19 de la 13. Introduzca el resultado. (Si es cero o menor, introduzca 0). Incluya esta cantidad en el total de la línea 7 del formulario 1040A. En el espacio de la izquierda de la línea 7 escriba «BCD». | 20 |

Nota: Si también reclama el crédito por cuidado de niños y dependientes, llene primero el formulario 1040A hasta la línea 20 y luego rellene las líneas 3-12 de la parte I.

Parte II

Nota: Si recibió un formulario 1099-INT o un formulario 1099-OID de una empresa de corretaje, introduzca el nombre de la empresa y el total de intereses que aparece en este formulario.

Parte III

Nota: Si recibió un formulario 1099-DIV de una empresa de corretaje, introduzca el nombre de la empresa y el total de dividendos que aparece en ese formulario.

Ingresos por intereses (vea pág. 24 de las instrucciones)	
Rellene esta parte y adjunte el régimen 1 al formulario 1040A si recibió más de 400\$ en intereses gravables.	
1 Lista de nombres de los pagadores	.Cantidad
	1
2 Sume las cantidades de la línea 1. Introduzca el total aquí y en el formulario 1040A, línea 8a.	2
Ingresos por dividendos (vea pág. 24 de las instrucciones)	
Rellene esta parte y acompañe el régimen 1 al formulario 1040A si recibió más de 400\$ en dividendos.	
1 Lista de nombres de los pagadores	Cantidad
	1
2 Sume las cantidades de la línea 1. Introduzca el total aquí y en el formulario 1040A, línea 9.	2

Figura 4.11. Formulario 1040A para la declaración individual del impuesto sobre la renta en Estados Unidos.

Figura 4.11.Bis (Cont.)

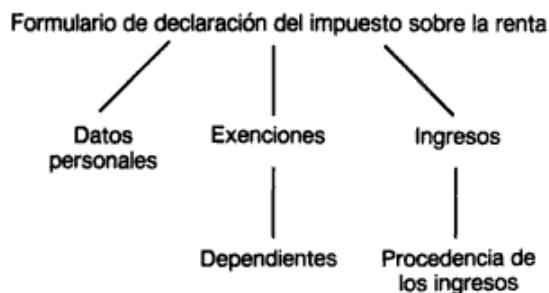


Figura 4.12. Árbol de áreas y subáreas del formulario mostrado en la figura 4.11.

bre y *Apellido* son *nombres únicos* de conceptos del formulario. En esa misma línea, *Ciudad* y *Pueblo* son los dos *nombres posibles* del concepto correspondiente. Para terminar, en la última línea, *Oficial* y *Soldado* indican los posibles *valores* para el concepto en cuestión. El esquema ER correspondiente debe tener cuatro atributos: *APELLIDO*, *NOMBRE*, *CIUDAD_DE_NACIMIENTO* y *RANGO_MILITAR*.

El texto estructurado, como *desde __/__/19__ hasta __/__/19__*, indica explícitamente la existencia de dos atributos, *COMIENZO_DEL_SERVICIO_MILITAR* y *FINAL_DEL_SERVICIO_MILITAR* y además proporciona información sobre la estructura de los datos (por ejemplo, necesidad de 6 bytes), que será útil en las siguientes fases del diseño.

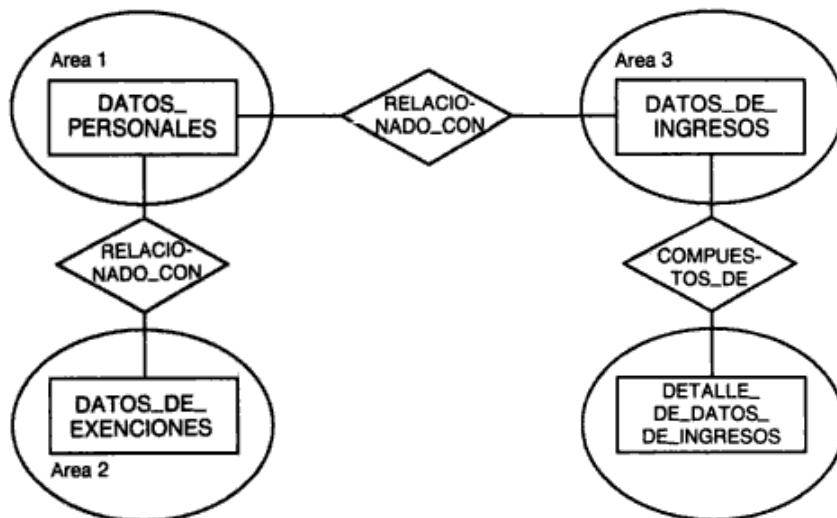


Figura 4.13. Esquema armazón de la base de datos para el formulario del impuesto sobre la renta.

Paso 2

Marque su estado civil declarado (sólo uno)	1 <input type="checkbox"/> Soltero. (Vea si puede usar el formulario 1040EZ.) 2 <input type="checkbox"/> Casado con declaración conjunta 3 <input type="checkbox"/> Casado con declaración separada. Introduzca arriba el número de seguro social del cónyuge y el nombre completo aquí. _____ 4 <input type="checkbox"/> Cabeza de familia (con persona calificada). (Vea la página 16.) Si la persona calificada es hijo... 5 <input type="checkbox"/> Viuda/o calificada/o con hijo...
--	---

Figura 4.14. Ejemplo de una lista.

Listas. En una lista se presentan exhaustivamente todos los posibles valores de un concepto; se selecciona algunos de ellos (por ejemplo, poniendo una marca) cuando se completa el formulario. La figura 4.14 muestra una lista del formulario de declaración del impuesto sobre la renta 1040A.

Cuando se traduce una lista a conceptos ER, es importante entender si las alternativas presentadas al usuario son *excluyentes*; en este caso, se puede utilizar un solo atributo con un valor sencillo para representar la lista de alternativas. Si, por el contrario, son posibles las *elecciones múltiples*, será necesario introducir un atributo sencillo con múltiples valores o bien un atributo para cada opción (de tipo booleano). En el ejemplo de la figura 4.14, las opciones son excluyentes; por esta razón, se introduce el atributo sencillo ESTADO_DE_DECLARACION.

Tablas. Las tablas se modelan convenientemente introduciendo entidades específicas que tienen dos conjuntos de atributos: los *identificadores* y los *valores*. Los identificadores seleccionan de manera inequívoca cada posición de la tabla, mientras que los valores corresponden al contenido de la tabla. En el formulario de declaración del impuesto sobre la renta 1040A, la segunda y tercera partes presentan una tabla cada una, las tablas de ingresos por intereses e ingresos por dividendos. En este caso, se refina la entidad DETALLE_DE_DATOS_DE_INGRESOS, presente en el esquema armazón, para dar dos entidades individuales: DETALLE_DATOS_INTERESES y DETALLE_DATOS_DIVIDENDOS, correspondientes cada una a una de las tablas; el identificador para cada fila de la tabla lo da la combinación del número de seguro social del declarante y el número de la fila. Los atributos de valor son NOMBRE_DEL_PAGADOR e IMPORTE.

Las tablas pueden ser más complejas (arrays multidimensionales); por ejemplo, la figura 4.15 muestra un ejemplo de array tridimensional que representa los gastos de una compañía durante un periodo de tres años. Cada gasto se refiere a un año, un mes del año y un periodo del mes. En el caso de los arrays multidimensionales, el número de atributos necesarios para la identificación es mayor. En este ejemplo, los atributos de identificación son: IDENTIFICADOR_DE_LA_COMPAÑIA, AÑO, MES y PERIODO; el atributo de valor es GASTO.

Partes opcionales del formulario. Un campo del formulario es opcional cuando se puede llenar o dejar vacío, dependiendo de reglas que usualmente

Gastos durante los tres últimos años														
	Año	Mes	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
Periodo	1988													
	1-15													
	16-31													
Periodo	1989		Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
	1-15													
	16-31													
Periodo	1990		Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
	1-15													
	16-31													

Figura 4.15. Ejemplo de tabla multidimensional.

aparecen en forma explícita, pero que también pueden ser implícitas. Por ejemplo, considérese el enunciado en la zona superior de la tercera parte del formulario del impuesto sobre la renta (Fig. 4.11): *Llene esta parte y acompañe el régimen 1 al formulario 1040A si recibió más de 400\$ en dividendos.* Esta oración indica que la tabla INGRESOS_POR_DIVIDENDOS entera puede dejarse vacía. Esta optionalidad se traduce al modelo ER asignando cero como cardinalidad mínima de la entidad DATOS_DE_INGRESOS en la interrelación DETALLE_DE_DIVIDENDOS.

La mayoría de las optionalidades se refieren a los atributos. Considérese de nuevo la parte del formulario de la figura 4.14 y obsérvese que las opciones 3 y 4 precisan llenar un espacio vacío. Esto corresponde a introducir algunos atributos adicionales: NUM_SEG_SOC_DEL_CONYUGE, NOMBRE_DEL_HIJO_PARA_CABEZA_DE_FAMILIA, y el atributo compuesto NOMBRE_COMPLETO_DEL_CONYUGE. Sin embargo, puesto que el relleno de espacios se requiere sólo en el caso de elecciones específicas, estos atributos son, ciertamente, opcionales (por ejemplo, con cardinalidad mínima de 0).

Datos derivados. Un campo contiene datos derivados cuando su valor se puede calcular a partir de otros datos del formulario. El formulario del impuesto sobre la renta contiene muchos ejemplos de datos derivados; por ejemplo, el número de casillas marcadas en el área de exenciones puede derivarse de las casillas individuales. Asimismo, los campos 11, 12c y 13 del área de ingresos

corresponden a datos calculados. Por último, los totales de las cantidades en las tablas para ingresos por intereses y por dividendos pueden derivarse sumando entradas individuales.

Es importante señalar que los datos derivados no se almacenan necesariamente en la base de datos, porque pueden ser calculados por un programa. Sin embargo, volver a calcularlos puede ser costoso; así, en algunas aplicaciones, los datos derivados se almacenan realmente dentro de registros de base de datos. Quizá lo más razonable en el nivel conceptual sea incluir los datos derivados e indicar con claridad cómo se pueden calcular.

La figura 4.16a muestra el esquema conceptual final. Todos los detalles del esquema se deben considerar con cuidado, especialmente los valores de cardinalidad de los atributos y los atributos compuestos. Nótese que los dependientes se modelan como una entidad en las subáreas de exenciones, puesto que se puede asociar algunas propiedades (atributos, interrelaciones, subentidades) con los dependientes. La figura 4.16b indica las reglas que se usan para calcular los datos derivados.

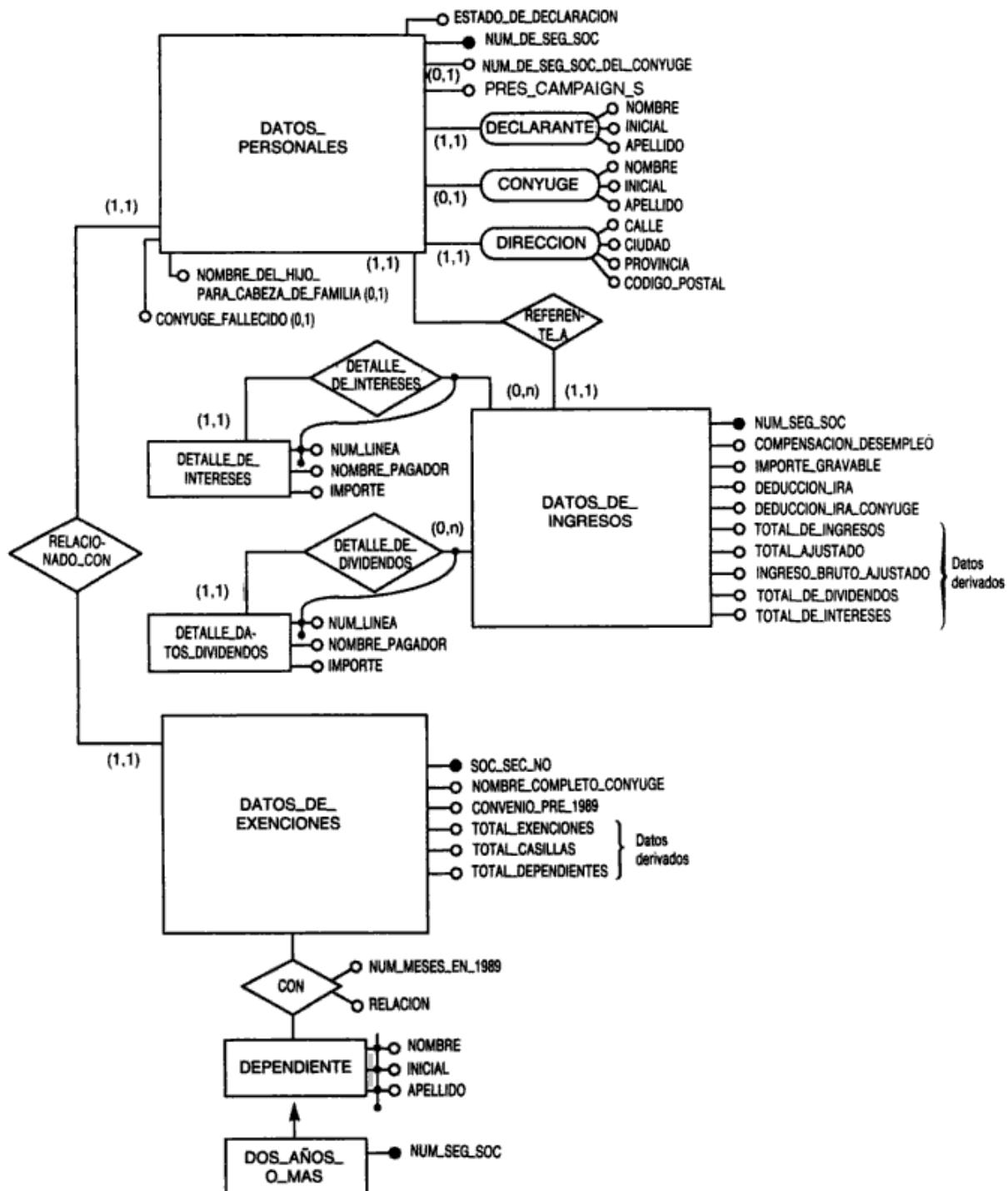
4.3. Diseño de vistas a partir de formatos de registros

Las aplicaciones comerciales implantadas en computadores usan invariablemente *archivos*, es decir, conjuntos de registros almacenados en la memoria secundaria. Cada registro consiste en un grupo de campos, los cuales pueden, a su vez, componerse de subcampos. En consecuencia, los registros suelen tener una estructura jerárquica y cada campo se sitúa en un determinado nivel dentro de la jerarquía. Los lenguajes más comúnmente usados para escribir aplicaciones son Cobol, PL/1, Fortran y C.

La estructura de los archivos se declara en los programas que los utilizan; por ejemplo, los archivos de Cobol se declaran en una determinada parte de los programas en Cobol, llamada división de datos (DATA DIVISION). La figura 4.17 muestra la declaración en Cobol de un archivo PEDIDO. Cada registro corresponde a un pedido; algunos campos (por ejemplo, CODIGO-DE-PIEZA, UNIDAD-DE-PIEZA, CANTIDAD) corresponden a unidades atómicas de información (campos elementales); otros campos (por ejemplo, VALOR, FECHA-DE-ENVIO) están a su vez estructurados en subcampos (campos compuestos).

En Cobol, como en otros lenguajes que utilizan archivos, varias cláusulas de la definición de archivos especifican el papel de los campos, su asignación de almacenamiento, los tipos de acceso disponibles para el archivo y otras características. Esta información es muy importante para determinar los significados de los campos, sus relaciones lógicas internas y las abstracciones definidas entre ellos, para así poder representar el archivo en términos de un esquema ER. Los programas de aplicación que no utilizan un DBMS (sistema de gestión de bases de datos) repiten, por lo regular, la definición de archivos en sus partes iniciales.

En el diseño de esquemas ER a partir de formatos de registros, se empieza



(a) Esquema conceptual final

Figura 4.16. Esquema del formulario de la declaración de la renta.

ATRIBUTO	DERIVACION
TOTAL DE INGRESOS	SALARIO TOTAL + INGRESOS POR INTERESES + INGRESOS POR DIVIDENDOS + COMPENSACION DESEMPLEO
TOTAL AJUSTADO	DEDUCCION IRA + DEDUCCION IRA CONYUGE
INGRESO BRUTO AJUSTADO	TOTAL DE INGRESOS – TOTAL AJUSTADO
TOTAL DE INTERESES	suma de IMPORTE en DETALLE DATOS INTERESES conectados por la interrelación DETALLE DE INTERESES
TOTAL DE DIVIDENDOS	suma de IMPORTE en DETALLE DATOS DIVIDENDOS conectados por la interrelación DETALLE DE DIVIDENDOS
TOTAL CASILLAS	1 + cardinalidad de NUM DE SEG SOC DEL CONYUGE
TOTAL DEPENDIENTES	cardinalidad de DEPENDIENTE
TOTAL EXENCIONES	TOTAL CASILLAS + TOTAL DEPENDIENTES

(b) Esquema conceptual final

Figura 4.16.Bis Esquema del formulario del impuesto sobre la renta (*continuación*).

introduciendo una entidad sencilla para representar el archivo y se le da el mismo nombre que al archivo. Esta elección es bastante natural, ya que un archivo es un conjunto de datos con la misma estructura. Luego, se consideran las partes

01 PEDIDO.

02 NUMERO-DE-PEDIDO.	PIC x(10).
02 FECHA-DE-EMISION.	
03 AÑO-DE-EMISION	PIC 9(2).
03 MES-DE-EMISION	PIC 9(2).
03 DIA-DE-EMISION	PIC 9(2).
02 FECHA-DE-ENTREGA.	
03 AÑO-DE-ENTREGA	PIC 9(2).
03 MES-DE-ENTREGA	PIC 9(2).
03 DIA-DE-ENTREGA	PIC 9(2).
02 VALOR.	
03 PRECIO	PIC 9(6)V99.
03 CODIGO-MONETARIO	PIC X(2).
03 TASA-DE-CAMBIO	PIC 9(6)V99.
02 LINEA-DE-PEDIDO OCCURS 10 TIMES.	
03 CODIGO-PIEZA	PIC 9(6).
03 CLAVE-DE-LINEA	PIC 9(3).
03 UNIDAD-DE-MEDIDA	PIC X(2).
03 CANTIDAD	PIC 9(6) COMPUTATIONAL.
02 CODIGO-DE-ALMACEN	PIC X(3).
02 CODIGO-DE-PROVEEDOR	PIC X(4).
02 CLIENTE	PIC X(15).
02 FABRICA	PIC X(2).

Figura 4.17. Descripción en Cobol de un archivo PEDIDO.

(cláusulas) de la definición del archivo para deducir propiedades estructurales adicionales del archivo. Así, la representación simple inicial del archivo se enriquece progresivamente con la introducción de nuevas entidades, interrelaciones, generalizaciones, atributos, etcétera. En este apartado se examinan algunas de las cláusulas que pueden aparecer en una definición de archivo y se ofrecen directrices generales para su transformación en elementos del modelo ER. Para concretar las ideas, se usará la terminología del lenguaje Cobol.

4.3.1. Campos simples

Un campo es simple cuando tiene una sola ocurrencia en cada caso de registro y es subindicado (o repetitivo) en caso contrario. Los campos simples pueden ser elementales o compuestos. Los campos elementales simples se traducen en atributos simples del modelo ER; los campos compuestos se traducen en atributos compuestos. Considérese el formato de registro de la figura 4.17. Las siguientes líneas se traducen en atributos simples de la entidad PEDIDO.

02 CLIENTE X(15).	
02 FABRICA	PIC X(2).

Las siguientes líneas se traducen en un atributo compuesto de la entidad PEDIDO.

02 FECHA-DE-EMISION.	
03 AÑO-DE-EMISION	PIC 9(2).
03 MES-DE-EMISION	PIC 9(2).
03 DIA-DE-EMISION	PIC 9(2).

4.3.2. Campos subindicados (repetitivos)

Los campos subindicados de Cobol tienen múltiples ocurrencias, y cada ocurrencia se identifica por un número progresivo. En Cobol, los campos subindicados están definidos en una cláusula OCCURS, la cual especifica el número de ocurrencias del campo en cada caso del registro. En la figura 4.17, LINEA-DE-PEDIDO es repetitivo y ocurre 10 veces en un registro de PEDIDO. Los campos subindicados con una cláusula OCCURS sencilla se traducen en atributos sencillos, y sus cardinalidades mínima y máxima corresponden al valor de la cláusula OCCURS.

Una estructura de datos usada con frecuencia en Cobol es la tabla, o array

CANTIDAD EN EXISTENCIA DE UN PRODUCTO

Mes	1983	1984	1985	1986
Ene	12	25	27	67
Feb	23	54	65	13
Mar	43	98	66	34
Abr	12	25	27	67
May	23	54	65	13
Jun	43	98	66	34
Jul	12	25	27	67
Ago	23	54	65	13
Sep	43	98	66	34
Oct	12	25	27	67
Nov	23	54	65	13
Dic	43	98	66	34

(a)

01 PRODUCTO

02 NOMBRE PIC x(20)
02 CODIGO PIC x(4)
02 PRECIO PIC 9(5)

02 TABLA-DE-CANTIDAD-EN-EXIST.

03 CANTIDAD-EN-EXIST-POR-AÑO OCCURS 4 TIMES.

04 CANTIDAD-EN-EXIST-POR-MES PIC 99 OCCURS 12 TIMES.

(b)

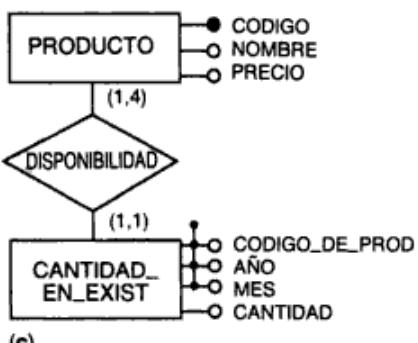


Figura 4.18. Tabla que muestra la cantidad en existencia de un producto, el formato de registro correspondiente y el esquema ER.

n-dimensional. Los arrays con n dimensiones se expresan en Cobol con el uso de n cláusulas OCCURS subordinadas. Los registros con más de una cláusula OCCURS subordinada se representan mejor introduciendo una nueva entidad.

Considérese la tabla de la figura 4.18a, que muestra la cantidad en existencia de un producto, clasificada por mes y año. Esta tabla se describe en Cobol como un campo subindicado, definido por el uso de dos casos de la cláusula OCCURS, como muestra la figura 4.18b. Como ya se expuso en el apartado anterior, una tabla se representa en el modelo ER introduciendo una entidad nueva; en este caso, se añade CANTIDAD_EN_EXIST que tiene como atributos de identificación CODIGO_DE_PROD, AÑO y MES, con el atributo de valor CANTIDAD. La entidad CANTIDAD_EN_EXIST está conectada a la entidad PRODUCTO por la interrelación DISPONIBILIDAD (Fig. 4.18c).

4.3.3. Redefinición de campos

La redefinición de campos permite a los programadores definir la misma parte de un registro usando diferentes cláusulas de definición de campos. Puede utilizarse con dos propósitos diferentes: 1) visualizar los mismos datos según diferentes puntos de vista, y 2) optimizar el espacio de almacenamiento físico.

La primera aplicación de la cláusula REDEFINES se muestra en la figura 4.19: el mismo conjunto de campos se agrega en dos grupos diferentes, de acuerdo con su uso en los procedimientos. El diseñador debe elegir la mejor representación conceptual, que puede ser cualquiera de las dos o una combinación de ellas. En este ejemplo, los subcampos BUSQUEDA se usan en un procedimiento de actualización que no distingue el nombre y el apellido, y que no requiere el día y el mes de nacimiento. En el esquema conceptual es preferible mencionar explícitamente todos los atributos, por lo cual se selecciona la primera alternativa.

La segunda aplicación de la cláusula REDEFINES usualmente indica la presencia de una jerarquía de generalización entre los conceptos descritos en el archivo. En la figura 4.20, el campo DATOS_DEL_TRABAJADOR se redefine dos veces en los campos DATOS_DE_SECRETARIA y DATOS_DE_DIRECTOR. Se puede traducir el archivo a un esquema con la entidad EMPLEADO y una jerarquía de generalización con las subentidades TRABAJADOR, SECRETARIA y DIRECTOR.

4.3.4. Punteros simbólicos

Un puntero simbólico es un campo de un registro que denota el identificador de otro registro. Los punteros simbólicos suelen usarse en Cobol para expresar interrelaciones lógicas entre archivos. Por ejemplo, en la figura 4.21 se definen tres formatos de registro, referentes a empleados, departamentos y proyectos. Las relaciones entre empleados, departamentos y proyectos en los que trabajan se expresan por medio de tres campos diferentes, que se usan como punteros: 1) CODIGO-DEPARTAMENTO liga los empleados con sus departamentos, 2) CODIGO-DEPARTAMENTO proyecto vincula los empleados con los proyectos en los que trabajan, y 3) CODIGO-DEPARTAMENTO liga los proyectos con los departamentos que los controlan.

```

01 PERSONA
02 DATOS-PERSONALES
03 NOMBRE
  04 APELLIDO          PIC x(20)
  04 NOMBRE-DE-PILA    PIC x(20)

03 FECHA-DE-NACIMIENTO
  04 AÑO               PIC 99
  04 MES                PIC 99
  04 DIA                PIC 99

02 DATOS-PERSONALES-BIS REDEFINES DATOS-PERSONALES
03 BUSQUEDA
  04 NOMBRE-S          PIC x(40)
  04 AÑO-S              PIC 99

03 FILLER              PIC 9(4)

```

(a)



(b)

Figura 4.19. Primer ejemplo de uso de la cláusula REDEFINES.

Los punteros se traducen a interrelaciones en el modelo ER. Estas interrelaciones conectan la entidad que tiene el campo puntero y la entidad correspondiente al archivo al que «apunta». Las cardinalidades mínima y máxima de las interrelaciones dependen del significado específico de los campos. La figura 4.21b muestra la traducción de los punteros recién mencionados a interrelaciones de las entidades EMPLEADO, DEPARTAMENTO y PROYECTO. Como caso particular, puede suceder que un puntero se refiera al campo identificador del registro en el que está definido; este caso se ejemplifica en la figura 4.22, que describe la estructura jerárquica de los proyectos y subproyectos desarrollados en una compañía. Estos punteros corresponden a las autointerrelaciones, o interrelaciones en *lazo* (es decir, las interrelaciones binarias de una entidad con la misma entidad).

4.3.5. Banderas

El término *bandera* hace referencia a campos o grupos de campos que suelen utilizar los programadores en Cobol para indicar si el campo o campos subse-

01 EMPLEADO.

02 CODIGO PIC x(7).
 02 TIPO-DE-TRABAJO PIC X.

02 DATOS-DEL-TRABAJADOR.

03 HORAS-SEMANALES PIC 99.
 03 EN-TURNO PIC X.
 03 AFILIACION-SINDICATO PIC X(6).

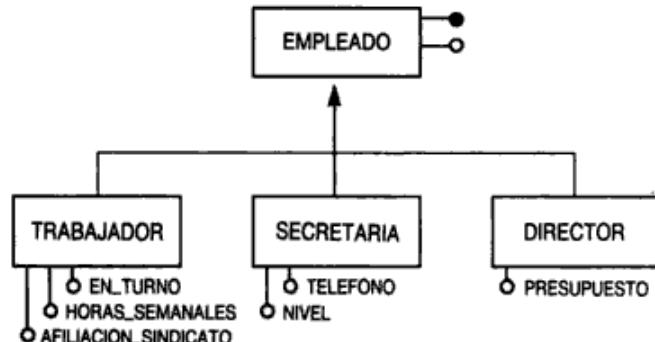
02 DATOS-DE-SECRETARIA REDEFINES DATOS-DEL-TRABAJADOR.

03 NIVEL PIC9.
 04 TELEFONO PIC 9(7).

02 DATOS-DE-DIRECTOR REDEFINES DATOS-DE-SECRETARIA.

03 PRESUPUESTO PIC 9(8).

(a)



(b)

Figura 4.20. Segundo ejemplo de uso de la cláusula REDEFINES.

cuentas en un caso de registro toman un valor o se dejan vacíos. Las banderas pueden indicar la presencia de un subconjunto entre dos (o más) conceptos distintos expresados en el archivo.

Como ejemplo, considérese en la figura 4.23 el archivo de pólizas de seguro de una compañía. Algunos campos son válidos para cualquier tipo de póliza (NUMERO, FECHA-DE-TERMINO, etc.); sólo algunas pólizas incluyen el riesgo de robo. Esta propiedad está puntuizada por el campo BANDERA_DE_ROBO: el valor es 0 para pólizas que cubren robo, 1 para las que no. Si el valor de BANDERA_DE_ROBO es 0, los campos CANTIDAD_ASEGURADA y COBERTURA deben especificarse. Nótese que ésta es una convención adoptada por el programador; sin embargo, en Cobol no se dispone de un control durante la ejecución para asegurar su cumplimiento. Se puede traducir la declaración de archivo de dos formas diferentes: 1) con una entidad única, en cuyo caso los campos a los que se refiere la bandera se traducen en términos de atributos con cardinalidad mí-

```

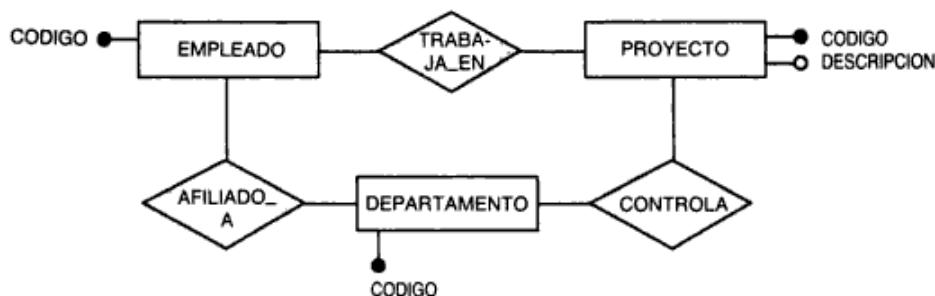
02 EMPLEADO.
  03 CODIGO          PIC x(10).
  03 CODIGO-DEPARTAMENTO   PIC x(5).
  03 CODIGO-PROYECTO    PIC x(7) OCCURS 10 TIMES.

02 DEPARTAMENTO.
  03 CODIGO PIC x(5).

02 PROYECTO.
  03 CODIGO      PIC x(7).
  03 COD_DEPT    PIC x(5).
  03 DESCRIPCION  PIC x(30).

```

(a)



(b)

Figura 4.21. Primer ejemplo de traducción de punteros.

nima opcional de cero, y 2) con dos entidades relacionadas por un subconjunto, como muestra la figura 4.23.

4.3.6. Reglas para valores de campos

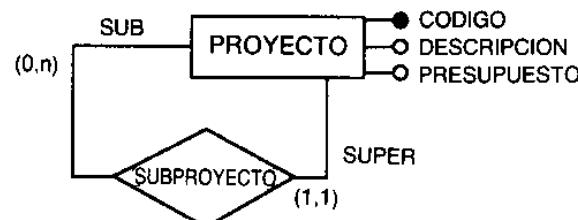
En muchos programas en Cobol, los usos específicos de los campos se expresan en términos de reglas basadas en los valores. Estas reglas son muy generales y típicamente incluyen intrincados trucos de programación, difíciles de entender. Dado que su semántica no se expresa mediante las propiedades estructurales de los datos, las reglas casi siempre se entienden con sólo estudiar los programas.

Como ejemplo, se muestra un archivo que trata de la contabilidad de una compañía. Se definen tres niveles de contabilidad: la división, el centro de costos y la cuenta específica. Las cuentas se agrupan por centro de costos, y éstos se agrupan por divisiones. Así, el campo CENTRO-DE-COSTOS (que indica el código de un centro de costos) sólo es significativo en registros de cuentas, y el campo DIVISION (que indica el código de una división) sólo es significativo en registros de centros de costos. El formato de registros del archivo se muestra en

01 PROYECTO.

02 CODIGO.	PIC x(5)
02 DESCRIPCION.	PIC x(30)
02 CODIGO-SUPERPROYECTO	PIC x(5)
02 PRESUPUESTO	PIC 9(9).

(a)



(b)

Figura 4.22. Segundo ejemplo de traducción de punteros.

la figura 4.24a. Los valores específicos de los códigos establecen si un caso de registro pertenece a uno de los tres niveles. Las reglas para los valores de los códigos se muestran en la figura 4.24b. Se puede concluir que: 1) el archivo es el resultado de la fusión de tres tipos lógicamente diferentes de registros, relacionados jerárquicamente, y 2) los campos CENTRO_DE_COSTOS y DIVISION se pueden considerar como punteros a otros registros del mismo archivo.

Partiendo de este análisis, en la figura 4.24c se muestra un posible esquema ER para la definición del archivo. La entidad INFORME_DE_CONTABILIDAD representa la raíz de una generalización cuyas subentidades son CUENTA, CENTRO_DE_COSTOS y DIVISION. Las dos interrelaciones de uno a muchos entre las subentidades expresan la estructura jerárquica definida entre los conceptos.

Este apartado se completa mostrando en la figura 4.25 el esquema ER que representa el archivo PEDIDO presentado en la figura 4.17. El archivo se traduce introduciendo dos entidades, PEDIDO y LINEA_DE_PEDIDO. Nótese que los dos atributos CODIGO_DE_ALMACEN y CODIGO_DE_PROVEEDOR son en realidad punteros, y pueden representarse de manera conveniente a través de interrelaciones con las entidades ALMACEN y PROVEEDOR.

4.4. Resumen

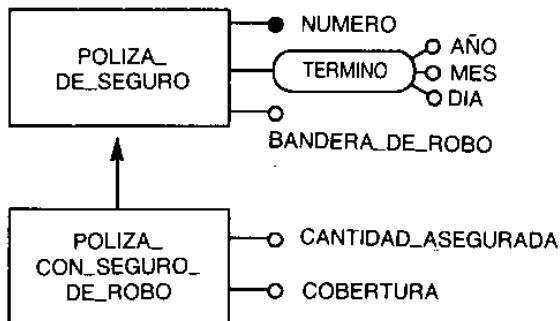
Este capítulo ha examinado tres enfoques diferentes para el diseño de esquemas conceptuales de bases de datos, con base en los distintos tipos de requerimientos de entrada. Se ha mostrado, casi siempre mediante reglas y ejemplos simples, cómo aprovechar la estructura de los requerimientos. Las sugerencias se utili-

```

01 POLIZA-DE-SEGURO.
02 NUMERO PIC X(10).
02 FECHA-DE-TERMINO.
  03 AÑO      PIC 9(2).
  03 MES      PIC 9(2).
  03 DIA      PIC 9(2).
02 BANDERA-DE-ROBO PIC9.
  88 NO-ROBO   VALUE 0.
  88 SI-ROBO   VALUE 1.
02 CANTIDAD-ASEGURADA      PIC 9(10).
02 COBERTURA            PIC 9(10).

```

(a)



(b)

Figura 4.23. Ejemplo de traducción de una bandera.

zan sobre todo durante la primera etapa del diseño, donde los requerimientos tienen que entenderse y corresponderse con los esquemas armazón conceptuales; después, el diseñador debe ser capaz de profundizar el análisis desarrollando los pasos de refinamiento de acuerdo con la metodología general expuesta en el capítulo 3.

Ejercicios

- 4.1. Construya un esquema conceptual para la siguiente descripción en lenguaje natural.

Diseñe un sistema de bases de datos para una Facultad de Farmacia, División de Farmacocinética Clínica. La división tiene proyectos de investigación en varias etapas de desarrollo: en curso, pendientes y completos. Cada proyecto obtiene fondos de una subvención única. Por lo general, la

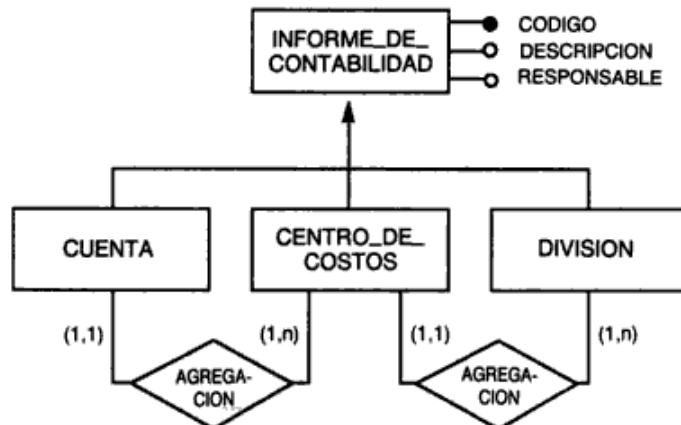
01 INFORME-DE-CUENTA.

02 CODIGO	PIC 9(4).
02 DESCRIPCION	PIC x(30).
02 CENTRO-DE-COSTOS	PIC 9(3).
02 DIVISION	PIC 9(2).
02 RESPONSABLE	PIC X(30).

(a) El archivo de contabilidad de una compañía

SI EL CODIGO ESTA ENTRE	EL REGISTRO SE REFIERE A
1000 y 9999	una cuenta
100 y 999	un centro de costos
1 y 99	una división

(b) Reglas definidas para el archivo de contabilidad



(c) Esquema conceptual

Figura 4.24. Ejemplo de traducción de archivos con reglas dependientes de los valores.

mayor parte de esta subvención se usa para pagar a los sujetos de estudio; los diversos fármacos y equipos usados en los experimentos suelen ser proporcionados por una o más compañías farmacéuticas. Un proyecto estudia los efectos del fármaco en varios sujetos, algunos de los cuales tienen regímenes terapéuticos desiguales. Varios empleados de la Facultad de Farmacia trabajan en cada proyecto, dirigido por un investigador principal; cada investigador principal puede controlar varios proyectos. Cuando se completa un estudio, se hace un informe de la investigación, donde se describen los resultados del estudio.

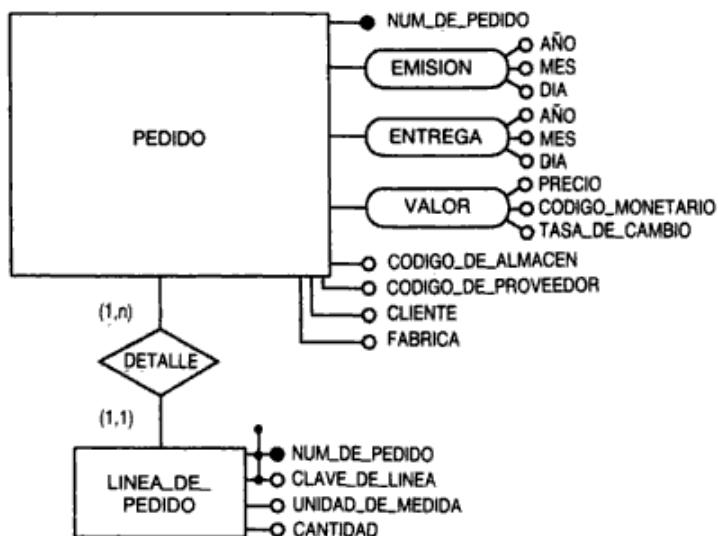


Figura 4.25. Representación ER del archivo PEDIDO.

- 4.2. Construya un esquema conceptual para la siguiente descripción en lenguaje natural.

Diseñe un sistema de bases de datos para controlar la información sobre rutas de una compañía de autobuses. Cada ruta cubierta por la compañía tiene un lugar de inicio y uno de término, pero puede pasar por varias paradas intermedias. La compañía está distribuida en varias sucursales. No todas las ciudades donde paran los autobuses tienen una sucursal; sin embargo, toda sucursal debe estar en una ciudad situada en las rutas de autobuses. Pueden existir múltiples sucursales en una misma ciudad y también múltiples paradas en la misma ciudad. La compañía asigna un autobús a cada ruta; algunas rutas pueden tener varios autobuses. Cada autobús tiene un conductor y un asistente, asignados al autobús por el día.

- 4.3. Construya un esquema conceptual para la siguiente descripción en lenguaje natural.

Diseñe la base de datos para la oficina de administración y reservas de una compañía de autobuses. Cada pasajero puede reservar un asiento para un tramo determinado de las rutas que cubre el autobús. Las rutas tienen un lugar de inicio, uno de término y varios lugares intermedios. Los pasajeros pueden especificar si prefieren la sección de fumadores o de no fumadores. Algunos pasajeros pueden abordar un autobús, aun sin reserva, cuando hay asientos vacíos. Con cada reserva, se almacenan el apellido, iniciales y número de teléfono del pasajero. En ocasiones no se realizan los viajes por malas condiciones atmosféricas, en cuyo caso se notifica a los pasajeros que tienen reservas. Al final del viaje, el asistente del conductor informa a la compañía la cantidad total de billetes comprados por los pasajeros en el

DEPARTAMENTO DE SALUD Y SERVICIOS DE REHABILITACION Informe del supervisor sobre investigación de accidentes Este formulario debe ser llenado por el supervisor después de un accidente laboral que resulte en lesiones				
1. Distrito	2. Nombre de la instalación		3. Unidad/Entidad	
5. Fecha del accidente		6. Hora	AM PM	7. Fecha de notificación
9. Nombre del accidentado		10. Número de seguro social		11. Tipo de trabajo
12. Clasificación de la lesión <input type="checkbox"/> 1. ^{er} auxilio <input type="checkbox"/> Incapacitación		13. Naturaleza de la lesión		14. Parte del cuerpo afectada
15. Origen de la lesión		16. Tipo de accidente		17. Condiciones de riesgo
18. Describa cómo ocurrió el accidente				
19. ¿Qué actos/omisiones y/o condiciones contribuyeron más directamente a este accidente?				
20. ¿Qué acción puede prevenir o controlar la repetición del accidente?				
Supervisor		Firma		Fecha
21. Acciones del Comité de Seguridad <input type="checkbox"/> Están de acuerdo con la acción correctiva del supervisor <input type="checkbox"/> No están de acuerdo con la acción correctiva. (¿Qué acción se tomaría?)				
Presidente		Firma		Fecha

Figura 4.26. Formulario para un informe de accidente.

Devuelva el formulario completo a:
DEPARTAMENTO DE PUBLICACIONES
Instituto de Ingenieros Eléctricos y Electrónicos
 345 East 47th Street
 New York, NY 10017

Las reimpresiones de artículos del IEEE pueden pedirse en lotes de 100 unidades a los siguientes precios. (Los autores cuyos nombres aparezcan en una revista con cargos por página, y que autoricen el pago de esos cargos, tienen derecho a 100 reimpressions gratis y pueden pedir 100 más al precio de «100 adicionales».)

Páginas	1-2	3-4	5-8	9-12	13-16	17-20	21-24	Cubiertas estándar	Cubiertas especiales
Primeras 100 (pedido mínimo)	53,35	88,95	148,70	209,70	252,85	297,25	358,25	82,65	106,70
100 adicionales	10,20	12,70	22,90	31,80	40,65	47,65	54,60	25,40	31,80

* Incluye título, autor(es), nombre(s) y línea de reimpresión.

Los cargos adicionales por envío distinto de correo ordinario, páginas especiales tituladas, material adicional o reimpressions en color, requerirán un presupuesto del impresor.

NO ENVIE PAGOS POR ADELANTADO.

Entrega en 30 días después de la fecha de publicación, aproximadamente.

Por favor, incluya una orden de compra, pedido o carta firmada por su agente de compras. Asegúrese de incluir cualquier pedido de sus coautores o de su organización.

TITULO Y AUTOR(ES) DEL ARTICULO _____

_____ NUM. _____

TRANSACCIONES/REVISTA _____ (MES Y AÑO, SI LOS CONOCE)

DIRECCION PARA FACTURACION

SI DESEA FACTURACION FRACCIONADA, INDIQUE LA CANTIDAD
Y ESCRIBA LA SEGUNDA DIRECCION AQUI

CANTIDAD: _____ CON CUBIERTAS ESTANDAR _____ CON CUBIERTAS ESPECIALES _____ TOTAL DE REIMPRESIONES _____
SIN CUBIERTAS (MUESTRA INCLUIDA) EN ESTE PEDIDO

LLENE LA ETIQUETA DEL IMPRESOR (ABAJO)
EN EL CASO DE PEDIDO FRACCIONARIO, INTRODUZCA LA SEGUNDA
DIRECCION ABAJO

SERVICIOS TECNICOS DE COMUNICACION
110 WEST 12TH AVENUE, NORTH KANSAS CITY, MO. 64116

SERVICIOS TECNICOS DE COMUNICACION
110 WEST 12TH AVENUE, NORTH KANSAS CITY, MO. 64116

CANT. _____ PEDIDO NUM. _____

CANT. _____ PEDIDO NUM. _____

A: _____

A: _____

Figura 4.27. Formulario para solicitar reimpressions de artículos del IEEE.

```

01 DATOS-DE-EMPLEADOS.
02 NUMERO-IDENTIFICACION          PIC 9(6).
02 NOMBRE                          PIC x(20).
02 EMPLEADO-SUPERVISOR           PIC x(20).
02 CONTROLADO-POR-MANDOS-DEL-SISTEMA    PIC x(5) OCCURS 10 TIMES.
02 CONTROLADO-POR-ALARMAS-DEL-SISTEMA    PIC x(5) OCCURS 10 TIMES.

01 MANDOS-DEL-SISTEMA.
02 TIPO                           PIC x(5).
02 FECHA                          PIC 9(6).
02 HORA..
  03 HORAS                         PIC 99.
  03 MINUTOS                        PIC 99.
  03 SEGUNDOS                       PIC 99.
02 UBICACION
  03 NOMBRE                         PIC x(20).
  03 LUGAR                           PIC x(20).
  03 TIPO                            PIC x(5).

01 ALARMAS-DEL-SISTEMA.
02 TIPO                           PIC x(5).
02 FECHA                          PIC 9(6).
02 VALOR                           PIC 9(8).
02 HORA..
  03 HORAS                         PIC 99.
  03 MINUTOS                        PIC 99.
  03 SEGUNDOS                       PIC 99.
02 UBICACION
  03 NOMBRE                         PIC x(20).
  03 LUGAR                           PIC x(20).
  03 TIPO                            PIC x(5).

01 COMUNICACIONES.
02 TECNOLOGIA                     PIC x(8).
02 VELOCIDAD                       PIC 9(6).
02 DISPOSITIVO-REMOTO OCCURS 10 TIMES.
  03 NOMBRE                         PIC x(10).
  03 TIPO                           PIC x(5).
  03 LUGAR                           PIC x(20).

02 DATOS-DE-CLABLEADO.
  03 MODEM.
    04 TIPO                          PIC x(10).
    04 VELOCIDAD-TRANSMISION        PIC 9(6).
    04 FILLER                         PIC x(20).
02 DATOS-DE-MICROONDAS REDEFINES DATOS-DE-CABLEADO.
  03 RADIO.
    04 RADIO                         PIC x(5).
    04 MODO-DE-TRANSMISION         PIC x(5).
    04 FRECUENCIA                    PIC x(10).
    04 ANTENA                        PIC x(5).

```

Figura 4.28. Definición de archivo del control distribuido de un proceso de manufactura.

autobús e indica esta cantidad a la oficina administrativa de la sucursal de destino de la ruta.

- 4.4. Construya un esquema conceptual para la siguiente descripción en lenguaje natural.

Diseñe la base de datos para un entorno de apoyo a la programación. En este entorno los programadores producen programas, que se escriben en determinados lenguajes de programación. Cada programa es escrito por un determinado programador, puede llamar a otros programas y puede ser utilizado por determinados usuarios. Los usuarios se reconocen por su nombre de entrada al sistema; los programadores se reconocen por su nombre de entrada al sistema y por su código; los programas poseen nombres compuestos que incluyen el nombre del programa, la extensión y el código del programador. Los programas tienen un número de versión, una fecha y una descripción breve; algunos programas interactúan con los DBMS. Cada DBMS mantiene datos almacenados en forma de relaciones, con varios atributos y una clave primaria. Cada base de datos la define un administrador de bases de datos, que es un programador especializado en la administración de datos.

- 4.5. Construya un esquema conceptual que contenga todos los datos del formulario mostrado en la figura 4.26, un informe sobre lesiones del Departamento de Salud y Servicios de Rehabilitación del Estado de Florida.
- 4.6. Construya un esquema conceptual que contenga todos los datos del formulario mostrado en la figura 4.27, un formulario de pedidos para solicitar reimpresiones de artículos del IEEE (Instituto de Ingenieros Eléctricos y Electrónicos).
- 4.7. Traduzca las definiciones de archivos en Cobol de la figura 4.28 a un esquema ER. Toda la aplicación trata del control distribuido de un proceso de manufactura. El archivo DATOS-DE-EMPLEADOS indica que cada empleado puede controlar mandos y alarmas. El archivo MANDOS_DEL_SISTEMA trata de los mandos disponibles para controlar procesos y los lugares donde cada mando puede accionarse. El archivo ALARMAS_DEL_SISTEMA trata de los mismos datos en el caso de las alarmas. Por último, el archivo COMUNICACIONES describe los datos sobre subsistemas de comunicaciones, su tecnología, los dispositivos conectados, ubicaciones de los dispositivos, etc. La transmisión de datos puede hacerse mediante microondas o por cable.

Bibliografía

P. P. Chen, «English Sentence Structure and Entity-Relationships Diagrams», *Information Sciences*, 29, 1983, 127-150.

Este artículo investiga la estrecha relación existente entre la estructura de la oración inglesa y los diagramas ER. Por ejemplo, los sustantivos corresponden a entidades y los verbos a interrelaciones.

V. de Antonellis y B. Demo, «Requirements Collection and Analysis». En S. Ceri, ed., *Methodology and Tools for Data Base Design*, North-Holland, 1983.

Se proponen algunas reglas empíricas para restringir la descripción en lenguaje natural de los requerimientos, de acuerdo con convenciones adecuadas (filtros para el lenguaje natural), y para calificar, dentro de las restricciones impuestas a las descripciones, enunciados que se refieren a datos, operaciones, sucesos y restricciones.

C. Elck y P. C. Lockemann, «Acquisition of Terminological Knowledge Using Database Design Techniques». En S. Navathe, ed., *Proc. ACM-SIGMOD International Conference*, Austin, Tex., 1985.

Este artículo propone conceptos, métodos y herramientas de apoyo para la construcción de una terminología comúnmente aceptada e integrada para usarse en un esquema conceptual.

J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.

Los esquemas ER no tienen el suficiente detalle para captar todo el significado del texto en inglés u otros lenguajes naturales. En su lugar, el autor propone el uso de grafos conceptuales que ofrecen una representación semántica intermedia: pueden representar el significado de las oraciones en inglés y, asimismo, pueden facilitar la traducción a diagramas ER.

M. Colombetti, G. Guida, y M. Somalvico, «NLDA: A Natural Language Reasoning System for the Analysis of Data Base Requirements». En S. Ceri, ed., *Methodology and Tools for Data Base Design*, North-Holland, 1983.

Este artículo presenta un sistema para el análisis de requerimientos en lenguaje natural que ayuda al diseñador a extraer de los requerimientos la descripción de los datos, transacciones y sucesos. Estos se filtran y clasifican adecuadamente y, en cierta medida, el sistema ayuda al diseñador a inferir la estructura del esquema de base de datos a partir de ellos.

M. Bouzeghoub y G. Gardarin, «The Design of an Expert System for Database Design», *Proceedings of the First International Workshop on New Applications of Databases*, Cambridge, Mass., 1983.

B. Flores, C. Proix y C. Rolland, «An Intelligent Tool for Information Design», *Proc. Fourth Scandinavian Research Seminar on Information Modeling and Data Base Management*, Ellivuori, Finlandia, 1985.

Estos dos artículos describen sistemas expertos para el diseño de bases de datos; entre otras características, los sistemas analizan, interpretan y estructuran la información proporcionada mediante enunciados en lenguaje natural. También interactúan con el usuario para eliminar las situaciones ambiguas o para pedir la información que falta. Final-

mente, se produce un modelo conceptual de la base de datos; el modelo es en realidad una rica red semántica. El primer enfoque se centra en las propiedades estáticas de los datos y utiliza la normalización; el segundo se centra en las propiedades dinámicas de los datos (por ejemplo, operaciones y sucesos).

C. Batini, B. Demo y A. di Leva, «A Methodology for Conceptual Design of Office Databases», *Information Systems*, 9, núms. 3, 4, 1984, 251-63.

Este artículo es la fuente de la metodología para derivar esquemas conceptuales que toma formularios como documentos de entrada (Apdo. 4.2).

K. H. Davis y K. Arora, «A Methodology for Translating a Conventional File System into an Entity-Relationship Model». En J. Liu, ed., *Proc. Fourth International Conference on Entity-Relationship Approach*, Chicago. IEEE Computer Society Press, 1985.

E. G. Nilsson, «The Translation of Cobol Data Structure to an Entity-Relationship Type Conceptual Schema», *Proc. Fourth International Conference on Entity-Relationship Approach*, Chicago, IEEE Computer Society Press, 1985.

Estos dos artículos describen metodologías para traducir un archivo convencional a un esquema ER. Las metodologías proceden a traducir primero la declaración del archivo convencional a un modelo de datos intermedio, eliminando los detalles sobre la implantación física del archivo. En seguida, el modelo intermedio se traduce al modelo ER.

H. Beck y S. B. Navathe, «Integrating Natural Language and Query Processing», *Proc. IEEE COMPCON*, San Francisco, feb.-mar. 1990.

Este artículo expone la correspondencia entre las consultas en lenguaje natural y una estructura conceptual en el modelo de datos CANDIDE, propuesto por los autores. Se desarrolla un léxico específico para el dominio de aplicación, y el razonamiento sobre los términos se utiliza usando las jerarquías conceptuales de cláusulas y atributos.

Integración de vistas

La integración de vistas es el proceso por el que se funden varios esquemas conceptuales en un *esquema conceptual global*, que representa todos los requerimientos de la aplicación. Ya vimos en el capítulo 3 que puede ser útil dividir los requerimientos de aplicaciones complejas en diferentes partes y luego continuar diseñándolas por separado. Esto puede ser una estrategia obligada cuando diferentes analistas participan en el proceso de diseño.

El principal objetivo de la integración de vistas es encontrar todas las partes de los esquemas conceptuales de entrada que se refieren a la misma porción de la realidad, y unificar su representación. Esta actividad se llama **integración de esquemas**; es muy compleja, ya que una misma parte de la realidad suele estar modelada de formas distintas en cada esquema.

También se necesita la integración en otro contexto, llamado **integración de bases de datos**, que implica la fusión de varias bases de datos distintas en una sola; en este caso, se debe construir primero un esquema conceptual de cada base de datos individual, y luego integrar esos esquemas. Esta actividad es necesaria en sistemas de información grandes, consistentes en varias bases de datos, y se expone en el apartado 7.3 (que trata sobre el desarrollo de los *diccionarios de datos*). Una aplicación especial de la integración de bases de datos ocurre en los *sistemas distribuidos de bases de datos*, en los que las bases de datos individuales se almacenan en diferentes computadores conectados en red. Los usuarios de estos sistemas deben contar con una visión unificada de la base de datos, que sea transparente con respecto a la distribución y asignación de datos. Aparte de afrontar problemas tecnológicos, los diseñadores de estos sistemas acaso necesiten integrar bases de datos existentes.

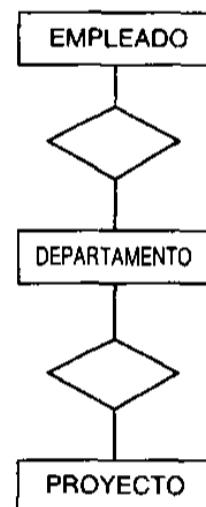
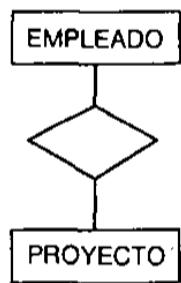
La organización de este capítulo es como sigue. En el apartado 5.1 se examinan los problemas y los factores que influyen en la actividad de integración. El apartado 5.2 describe la integración *a gran escala*, es decir, cómo organizar la integración de varios esquemas. Los apartados siguientes tratan sobre la integración *a pequeña escala*, es decir, entre dos esquemas de entrada. El apartado 5.3 se refiere al análisis de los conflictos y su resolución, y el apartado 5.4 trata la fusión de vistas.

5.1. Aspectos de la integración de vistas

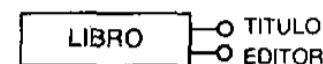
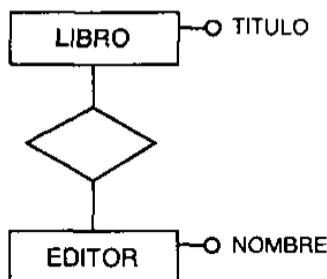
La mayor dificultad de la integración de vistas radica en descubrir las diferencias entre los esquemas que se van a fusionar. Las diferencias de modelado se deben a las siguientes causas.

Perspectivas diferentes. En el proceso de diseño, los diseñadores modelan los mismos objetos desde su propio punto de vista. Los conceptos pueden enfocarse desde distintos niveles de abstracción o representarse utilizando propiedades diferentes. Se da un ejemplo en la figura 5.1a: la relación entre EMPLEADO y PROYECTO se percibe como una interrelación en un esquema, y como la combinación de dos interrelaciones, en otro.

Equivalencia entre las construcciones del modelo. Los modelos conceptua-



(a) Perspectivas diferentes



(b) Construcciones equivalentes

Figura 5.1. Causas de la diversidad de esquemas.

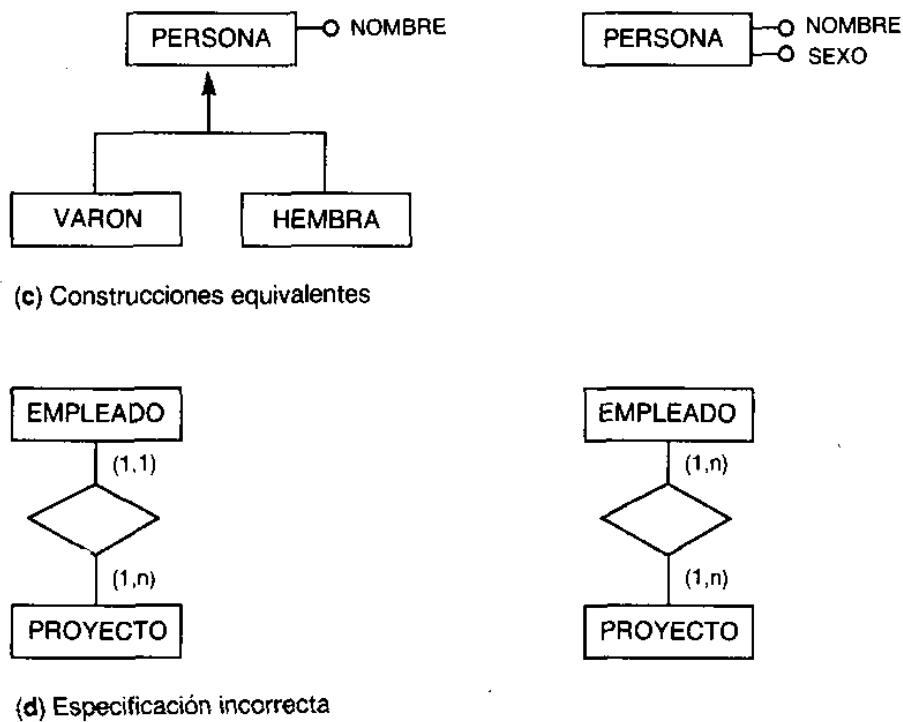


Figura 5.1.Bis Causas de la diversidad de esquemas (*continuación*).

les tienen una rica variedad de estructuras de representación; por tanto, permiten distintas representaciones equivalentes de la misma realidad. Por ejemplo, en la figura 5.1b, la asociación entre libro y editor se representa en un esquema mediante una interrelación de las entidades LIBRO y EDITOR, y como un atributo de la entidad LIBRO en el otro esquema. De forma similar, en la figura 5.1c, la partición de las personas en varones y hembras se representa con una jerarquía de generalización entre las entidades PERSONA, VARON y HEMBRA, en un esquema, y por el atributo SEXO de la entidad PERSONA, en el otro.

Especificaciones de diseño incompatibles. Los errores durante el diseño de vistas relacionados con nombres, estructuras y restricciones de integridad pueden producir entradas erróneas para la actividad de integración. Tales errores se deben detectar y corregir durante la integración. Por ejemplo, en la figura 5.1d, el primer esquema indica que cada empleado siempre se asigna a un solo proyecto; no obstante, el segundo esquema indica que cada empleado trabaja en múltiples proyectos. Los dos esquemas parecen correctos, pero uno de ellos es erróneo.

Cada una de estas causas puede dar como resultado un **conflicto**, es decir, representaciones diferentes de los mismos conceptos. La presencia de conflictos que se influyen mutuamente hace que el proceso de integración no sea trivial. El enfoque de este libro se muestra en la figura 5.2. Durante el **análisis de conflictos** se buscan los conflictos; el enfoque resalta la importancia de una pronta

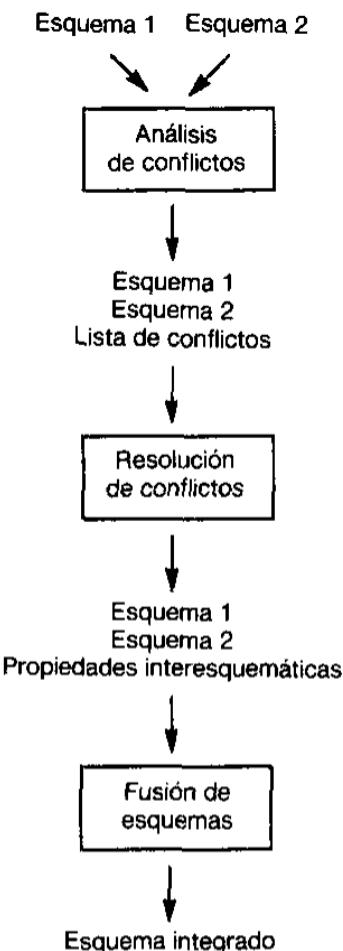


Figura 5.2. El enfoque de este libro hacia la integración de visitas.

identificación de los conflictos. Durante la **resolución**, se modifican uno de los esquemas, o ambos, para resolver o eliminar cada conflicto; esto debe hacerse en consulta con el diseñador. Durante la **fusión de esquemas**, se superponen los esquemas y se obtiene un esquema preliminar integrado. Puesto que todos los conflictos ya han sido resueltos, esta actividad es muy simple.

5.2. Integración de vistas a gran escala

En los proyectos grandes de diseño de bases de datos, es bastante común producir decenas o incluso cientos de esquemas que deben ser integrados. Este proceso requiere establecer una disciplina para seleccionar una secuencia apropiada de integraciones individuales de vistas. El enfoque más general del proceso de integración se muestra en la figura 5.3. Dicho enfoque procede con la inte-

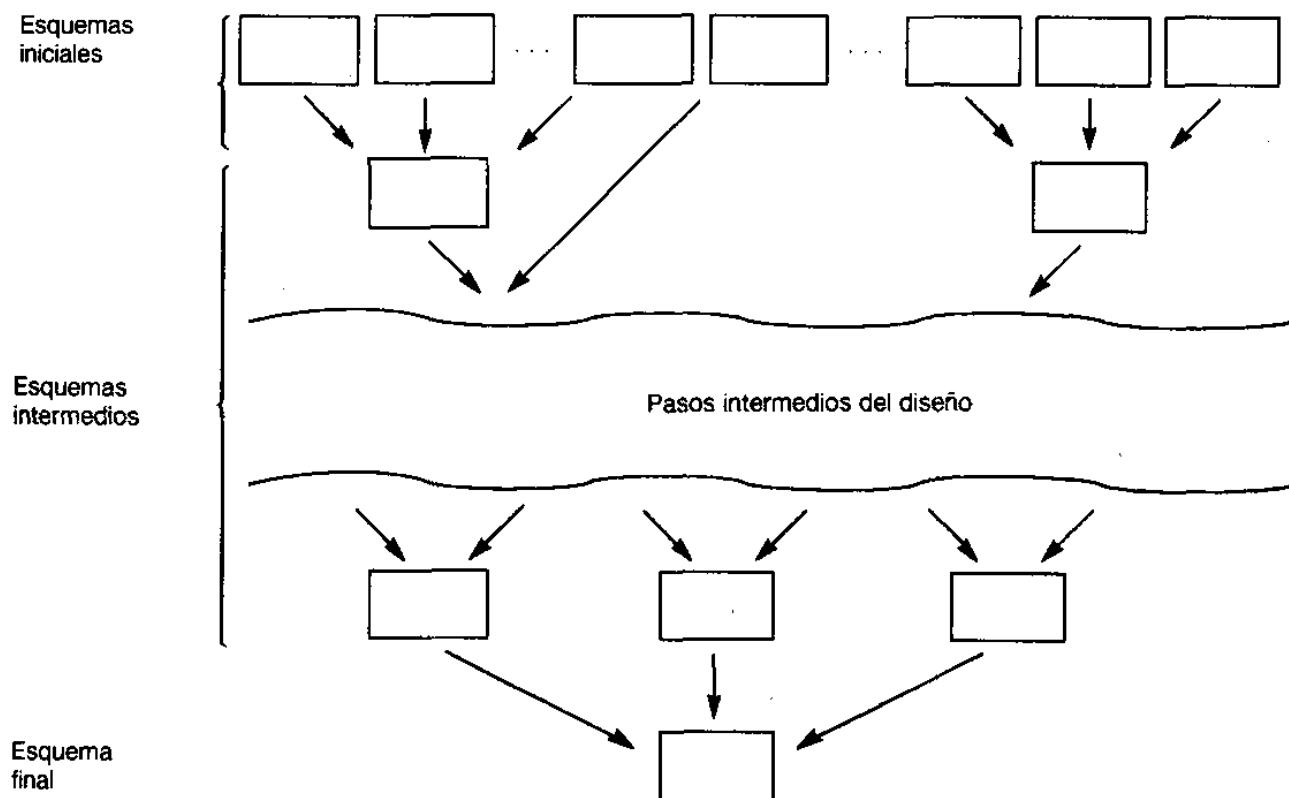


Figura 5.3. El enfoque más general hacia la integración de vistas.

gración de varios esquemas a la vez y, por tanto, abarca varios esquemas co-existentes y parcialmente integrados.

La integración de varios esquemas al mismo tiempo no es muy conveniente, porque resulta bastante difícil descubrir los conflictos. Se sugiere, en cambio, considerar sólo un par de esquemas a la vez; se sugiere también que los resultados de la integración de esquemas se acumulen en un esquema único, que evolucione gradualmente hacia el esquema conceptual global. Este enfoque hacia la integración de vistas se muestra en la figura 5.4: un esquema se amplía progresivamente para incluir nuevos conceptos de otros esquemas.

Primero, se debe elegir el orden de las comparaciones de esquemas. Si el proceso de integración se desarrolla de acuerdo a una estrategia de diseño mixta (descendente y ascendente), como se describió en el apartado 3.2.4, el *esquema armazón* debe elegirse como entrada del primer proceso de integración. Los esquemas deben irse agregando progresivamente al esquema armazón que funciona como eje del procedimiento. El orden en que se consideran los otros esquemas no tiene especial importancia. Si no se dispone de un esquema armazón, el diseñador debe establecer prioridades entre los esquemas, basándose en su importancia, compleción y fiabilidad. El esquema más importante, llamado *esquema de gestión*, debe considerarse en el primer paso de integración. Una clara ventaja de esta política es que primero se logra la integración de los esquemas

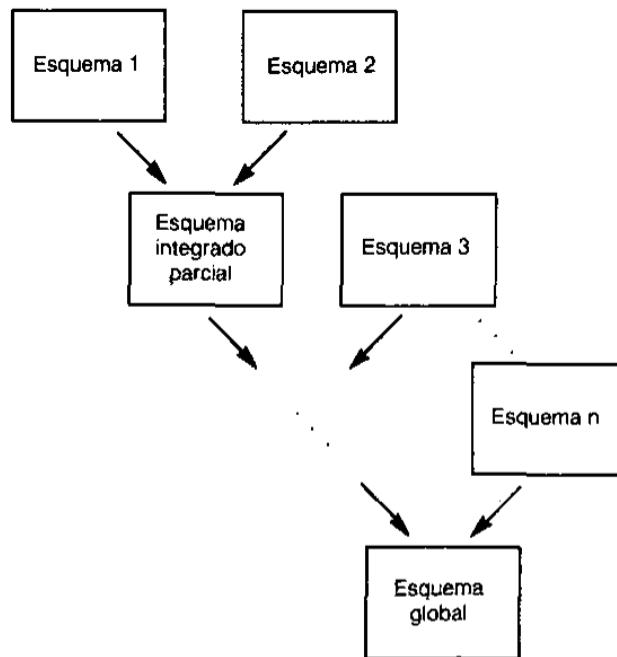


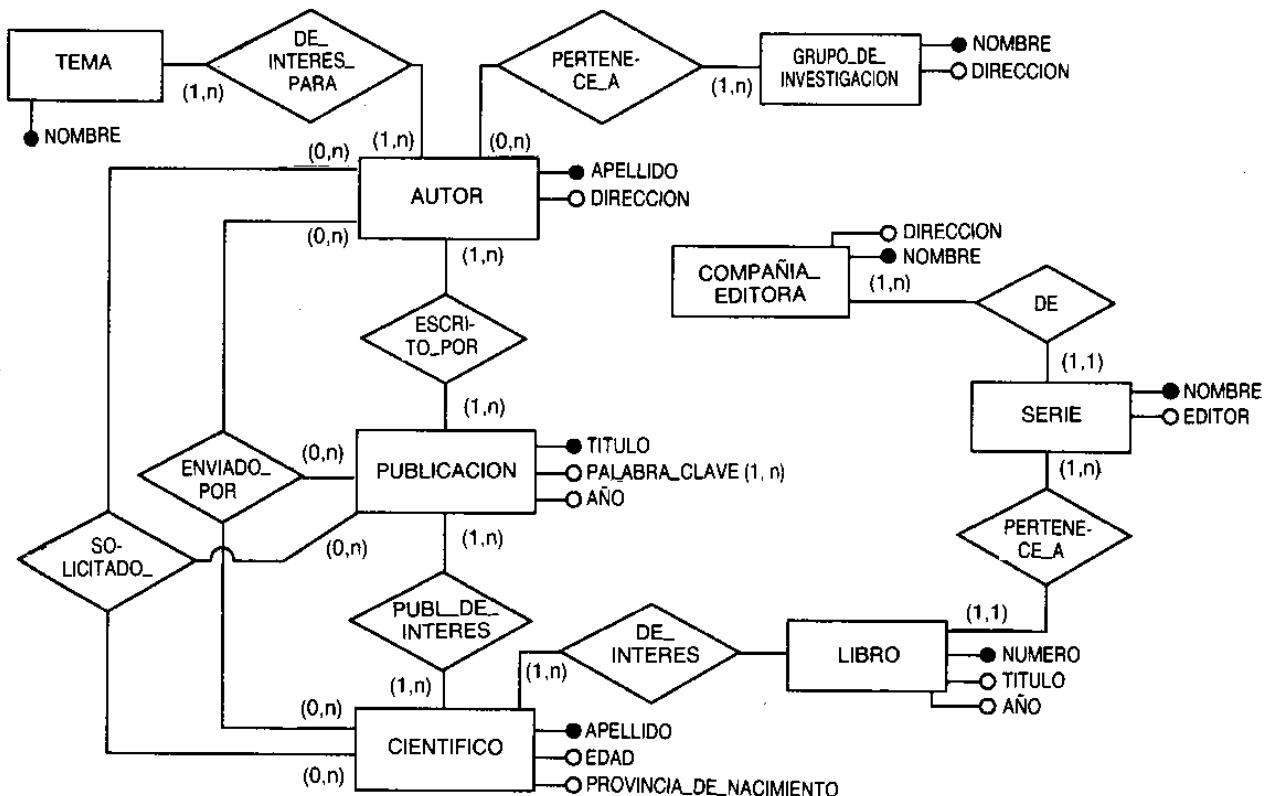
Figura 5.4. La secuencia sugerida para las actividades de integración de vistas.

más relevantes, lo que lleva a una mejor convergencia y estabilidad del proceso de diseño. Además, cuando hay necesidad de resolver conflictos, la solución se basa en las vistas más importantes.

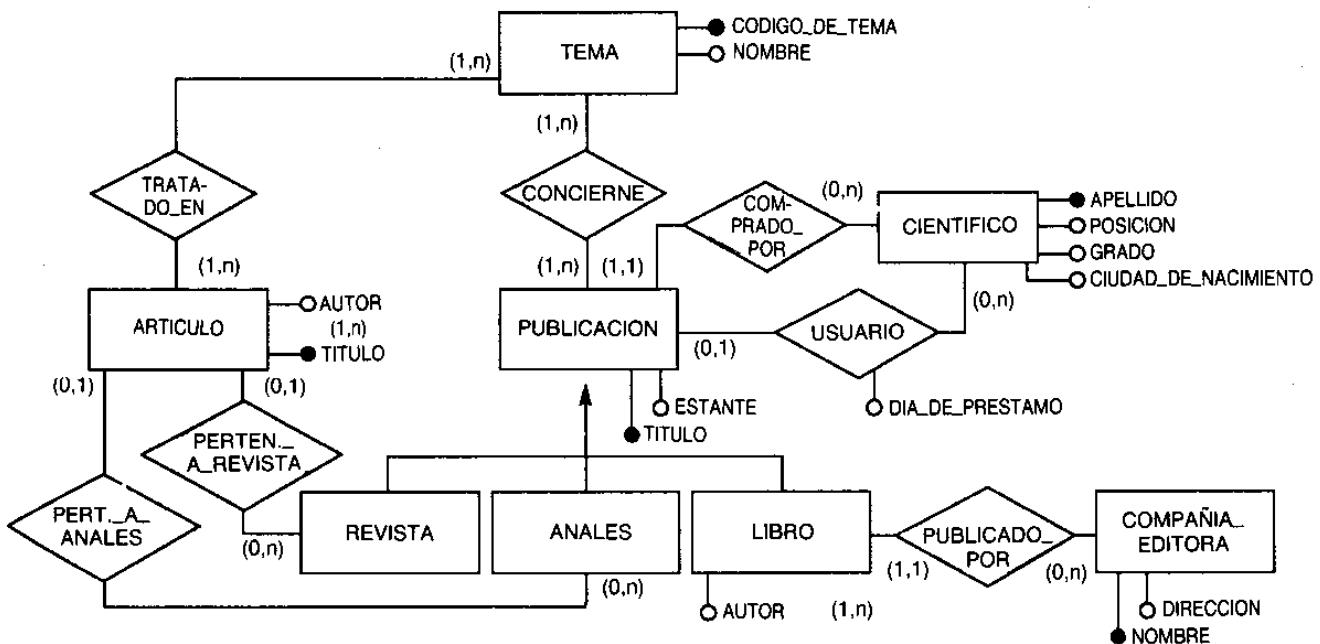
5.3. Análisis y resolución de conflictos

Se concentrará ahora la atención en el proceso de integración *a pequeña escala*, esto es, entre un par de esquemas, utilizando un ejemplo relativo a la administración de una biblioteca. Los esquemas de entrada se muestran en la figura 5.5. El esquema del científico (esquema 1) describe la estructura de la biblioteca privada de un investigador. El esquema del bibliotecario (esquema 2) describe la estructura de la biblioteca central de un departamento. Las tablas 5.1 y 5.2 describen los conceptos de los esquemas 1 y 2 cuyos significados no son obvios.

El análisis de conflictos intenta detectar todas las diferencias en la representación de la misma realidad en los dos esquemas. Pueden distinguirse dos tareas principales: 1) **análisis de conflictos de nombres**, en el cual los nombres de los conceptos de los esquemas se comparan y unifican, y 2) **análisis de conflictos estructurales**, en el que las representaciones de los conceptos de los esquemas se comparan y unifican.



(a) Esquema del científico (esquema 1)



(b) Esquema del bibliotecario (esquema 2)

Figura 5.5. Esquemas de entrada para la integración de vistas.

Tabla 5.1. Conceptos del esquema 1 (esquema del científico)

Nombre	Descripción
Autor	Los autores de las publicaciones de interés para los científicos.
Publicación	Las publicaciones guardadas por los científicos en sus estantes privados; los científicos las obtienen, por lo general, directamente de los autores.
Tema	Las áreas de investigación de interés para los autores.
Solicitado por	Conecta los artículos solicitados por algún científico con el autor a quien ha sido hecha la solicitud.
Enviado por	Conecta los artículos que han sido enviados por los autores con los científicos que los han solicitado.

5.3.1. Análisis de conflictos de nombres

Existen dos fuentes de conflictos de nombres: los sinónimos y los homónimos. Se dan **sinónimos** cuando los mismos objetos del dominio de aplicación se representan con diferentes nombres en los dos esquemas; los **homónimos** se dan cuando se representan diferentes objetos del dominio de aplicación con el mismo nombre en los dos esquemas. Para descubrir los sinónimos y homónimos, el diseñador se guía por semejanzas o discrepancias entre los conceptos, que pueden sugerir la presencia de un conflicto de nombres. La **semejanza de conceptos** surge cuando conceptos con nombres diferentes poseen algunas propiedades y restricciones comunes en los esquemas. La semejanza de dos conceptos indica que pueden ser sinónimos. La **discrepancia de conceptos** surge cuando conceptos con el mismo nombre poseen diferentes propiedades y restricciones en los esquemas. La discrepancia entre dos conceptos indica que pueden ser homónimos.

Los términos *propiedades* y *restricciones*, en estas definiciones, se definen como sigue. Las **propiedades** de un concepto son todos los demás *conceptos vecinos* en el esquema. Por ejemplo, las propiedades de una entidad dada son todos sus atributos, así como las interrelaciones, subconjuntos y generalizaciones

Tabla 5.2. Conceptos del esquema 2 (esquema del bibliotecario)

Nombre	Descripción
Publicación	Las publicaciones que actualmente se conservan en la biblioteca.
Artículo	Los artículos publicados en revistas o anales guardados en la biblioteca.
Tema	Temas de los artículos.
Comprado por	Indica qué científico es responsable de la subvención usada para adquirir la publicación.

Tabla 5.3. Propiedades vecinas y restricciones

Elemento del esquema	Propiedades vecinas	Restricciones
Entidad	Sus atributos, interrelaciones adyacentes, subconjuntos y jerarquías de generalización.	Cardinalidades mínima y máxima de las interrelaciones donde la entidad participa; identificadores.
Interrelación	Sus atributos; entidades participantes.	Cardinalidades mínima y máxima de las entidades participantes.
Atributos	Las entidades o interrelaciones a las que pertenecen.	Cardinalidades mínima y máxima, conjunto de valores, identificadores que incluyen el atributo.

en los que participa. Las **restricciones** son reglas o condiciones que limitan el conjunto de casos válidos del esquema. Estas incluyen, por ejemplo, las restricciones de cardinalidad para las interrelaciones y las generalizaciones. La tabla 5.3 muestra las propiedades vecinas y las restricciones para las entidades, interrelaciones y atributos.

Como consecuencia de la detección de conceptos semejantes y discrepantes, se puede efectuar en los esquemas algunas modificaciones. A todas las posibles modificaciones las llamamos *guiones de modificación*. Los guiones de modificación para un conflicto de nombres suponen cambiar el nombre del concepto, y pueden también implicar la adición de alguna *propiedad interesquemática*. A continuación se explican estas dos ideas.

Se **cambia el nombre de un concepto** siempre que se detecta un sinónimo o un homónimo. Los sinónimos deben eliminarse para evitar la ambigüedad. Por ejemplo, cuando dos conceptos como USUARIO y CLIENTE son sinónimos, se selecciona uno de ellos, por ejemplo USUARIO, y se cambia el nombre de CLIENTE a USUARIO. Para el caso de un homónimo, supóngase que REGISTRO, en una perspectiva, se refiere al proceso de registro de una persona que alquila un coche, mientras que, desde otro punto de vista, significa hacer una reserva de un coche. En este caso, el nombre del concepto desde la segunda perspectiva se debe cambiar a RESERVA.

Las **propiedades interesquemáticas** expresan las restricciones mutuas entre conceptos que aparecen en diferentes esquemas. Por ejemplo, la entidad CANDIDATO_DOCTOR en una perspectiva puede estar limitada a ser un subconjunto de la entidad ESTUDIANTE en otra perspectiva. Estas propiedades deben anotarse como extensiones de los dos esquemas; se utilizarán posteriormente en la fusión de los esquemas.

La figura 5.6 muestra un ejemplo de una posible modificación por un con-

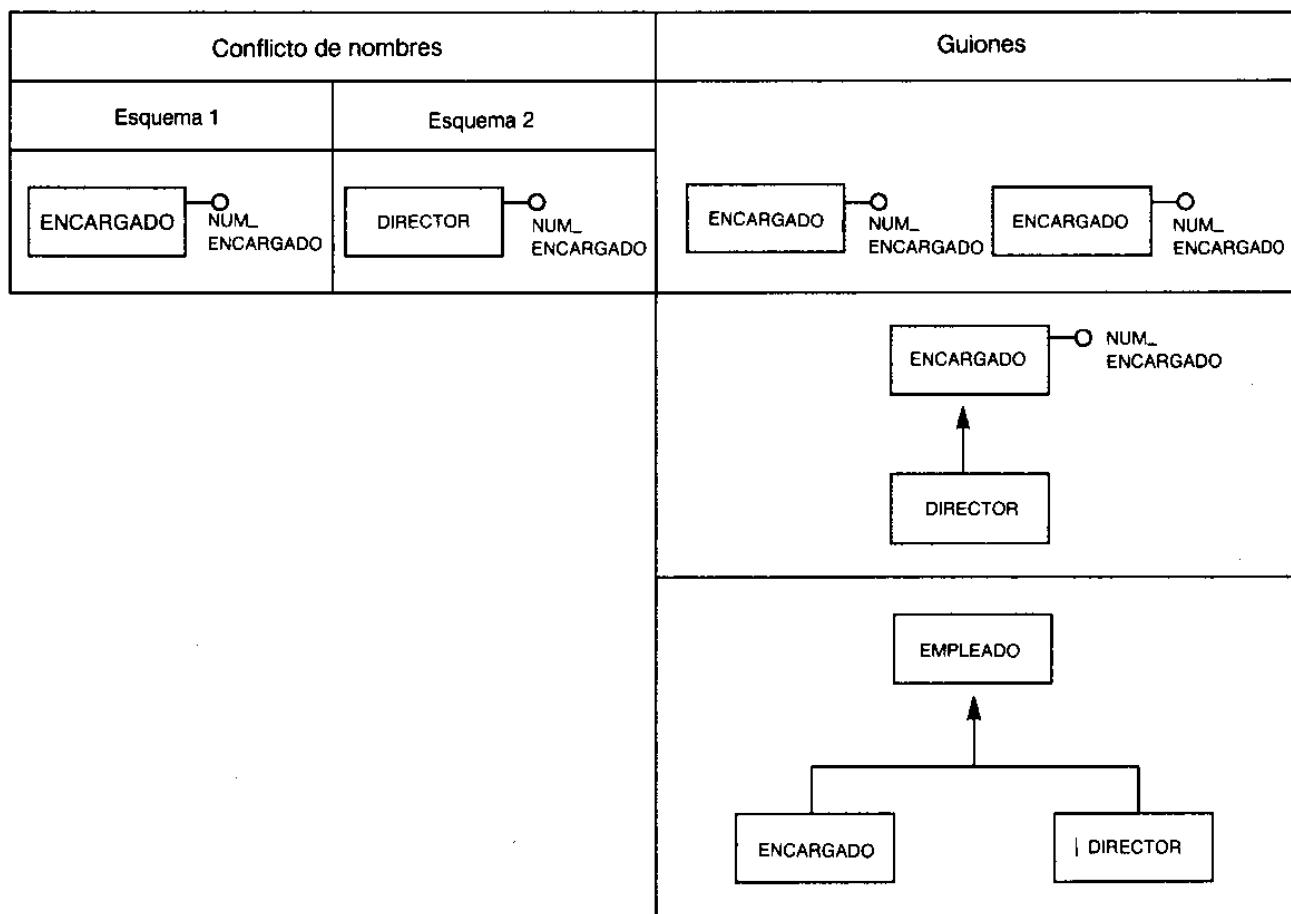


Figura 5.6. Ejemplos de escenarios para modificar un conflicto de nombres.

flicto de nombres. El esquema 1 contiene la entidad ENCARGADO; el esquema 2 contiene la entidad DIRECTOR. En el primer guión, los nombres se consideran sinónimos, y se cambia el nombre en el segundo esquema. En el segundo guión, los nombres se refieren a conceptos diferentes, relacionados por un subconjunto. En el tercer guión, los nombres se refieren a conceptos diferentes, relacionados por una jerarquía de generalización con un tercer concepto, más general. El subconjunto y la jerarquía de generalización son ejemplos de propiedades interesquemáticas.

En los esquemas que se representan en la figura 5.5 pueden encontrarse ejemplos de semejanza y de discrepancia entre conceptos.

1. La entidad TEMA aparece en los dos esquemas, pero con propiedades diferentes. En el esquema 1, TEMA se refiere al interés de un autor, mientras en el esquema 2 se refiere a lo que tratan las publicaciones. Este es un ejemplo de discrepancia de conceptos.
2. La entidad PUBLICACION se refiere, en el primer esquema, a un objeto soli-

citado a los autores y enviado por ellos. En el segundo esquema, se refiere a un objeto comprado (o solicitado en préstamo) por un científico. Esto es también un ejemplo de discrepancia.

3. El atributo PALABRA_CLAVE de la entidad PUBLICACION, en el primer esquema, y la entidad TEMA en la interrelación CONCIERNE con PUBLICACION, en el segundo esquema, tienen las mismas cardinalidades mínima y máxima. Esto es un ejemplo de semejanza de conceptos.

Los homónimos en los casos 1 y 2 se resuelven cambiando los nombres de TEMA y PUBLICACION en el primer esquema a AREA_DE_INVESTIGACION y ARTICULO, respectivamente. Nótese asimismo que el nombre de la interrelación PUBL_DE_INTERES, en el primer esquema, debe entonces cambiarse a ARTICULO_DE_INTERES. Se evitan los sinónimos en el caso 3 cambiando el nombre de PALABRA_CLAVE a TEMA en el primer esquema.

No todos los conflictos de nombres se pueden descubrir usando semejanza de conceptos o discrepancia de conceptos. Por ejemplo, considérese la entidad CIENTIFICO, que en el esquema 1 desempeña el papel de conseguir e interesarse por artículos y libros; en cambio, en el esquema 2 participa en la compra de publicaciones. A pesar de la discrepancia, las dos entidades sí corresponden al mismo concepto de la realidad.

5.3.2. Análisis de conflictos estructurales

Después de realizar un análisis de los conflictos de nombres, se consigue la *unificación de los nombres*. En esta situación, se supone que dos conceptos (atributos, entidades o interrelaciones) con el mismo nombre representan el mismo concepto de la realidad. Durante el análisis de conflictos estructurales, se comparan los conceptos que tienen el mismo nombre en los esquemas de entrada, para ver si pueden fusionarse. Se utilizan las siguientes categorías:

1. Conceptos *idénticos*; que poseen exactamente la misma estructura de representación y propiedades vecinas.
2. Conceptos *compatibles*; que poseen diferentes estructuras de representación o propiedades vecinas no contradictorias. Por ejemplo, el uso de una entidad y un atributo para representar el mismo concepto (Fig. 5.1b) es un ejemplo de compatibilidad de conceptos. Los conflictos de compatibilidad se resuelven con facilidad alterando una de las dos representaciones.
3. Conceptos *incompatibles*; que poseen propiedades contradictorias. Las fuentes de incompatibilidad deben eliminarse antes de fusionar los esquemas. Algunas de las posibles incompatibilidades en el modelo ER son:
 - a) *Cardinalidades diferentes* para el mismo atributo o entidad.
 - b) *Identificadores diferentes*: un identificador en un esquema no es un identificador en el otro.

- c) *Interrelaciones inversas de subconjuntos*: la entidad A es subconjunto de la entidad B en un esquema, y B es subconjunto de A en el otro esquema.

Las posibles soluciones de la incompatibilidad incluyen la selección de una representación en vez de otra y la construcción de una representación común tal que todas las restricciones de los dos esquemas estén incluidas en el esquema integrado.

Considérense de nuevo los esquemas del bibliotecario y del científico para buscar conceptos compatibles e incompatibles. Dos conceptos compatibles son:

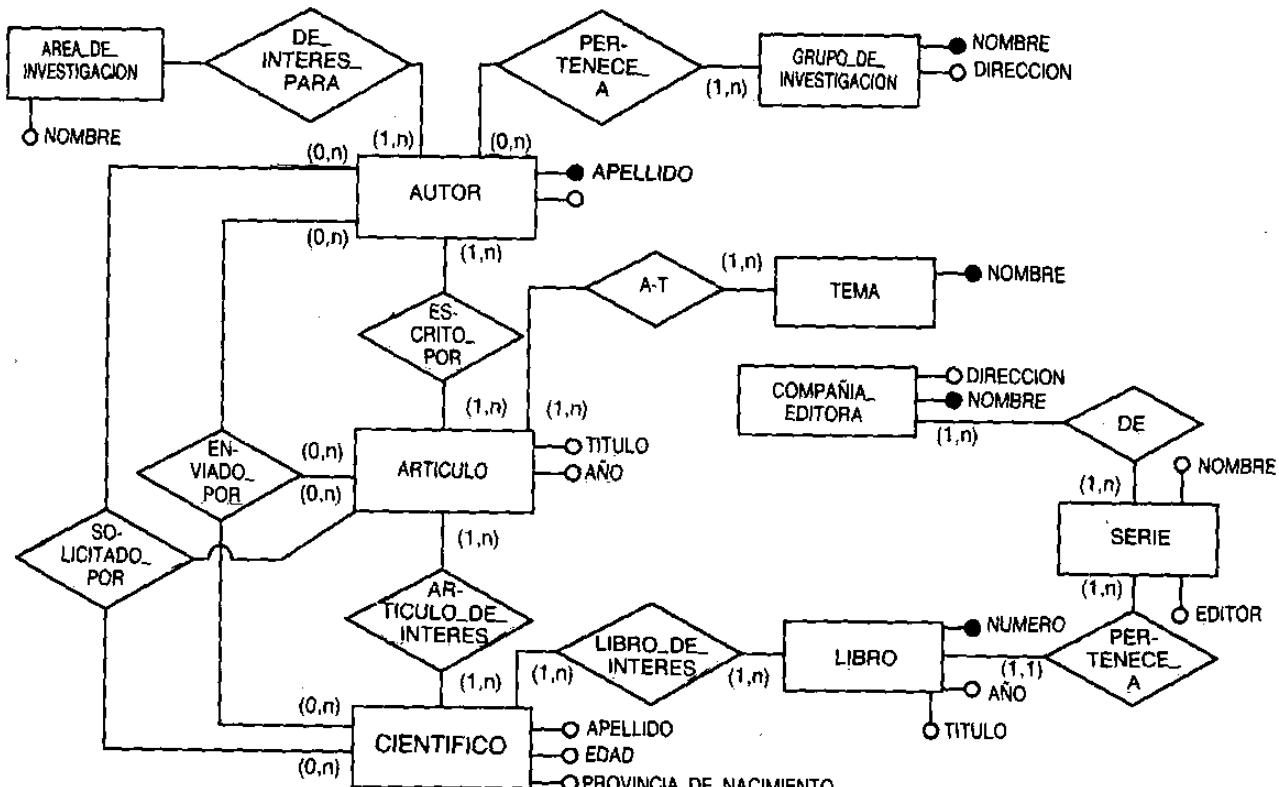
1. CIENTIFICO, que es una entidad en los dos esquemas, aunque posee diferentes atributos e interrelaciones. Representa el mismo objeto y no necesita ninguna actividad de reestructuración.
2. AUTOR y TEMA, que poseen diferentes estructuras de representación en los dos esquemas. AUTOR debe transformarse en una entidad en el esquema 2 y TEMA debe transformarse en una entidad en el esquema 1. En los dos casos se introducen interrelaciones adecuadas.

Surgen dos casos de incompatibilidad de conceptos. El primero se refiere a la cardinalidad mínima de la entidad COMPAÑIA_EDITORA, en la interrelación PUBLICADO_POR con LIBRO: es 0 en el esquema 2, pero es 1 (a través de la entidad SERIE) en el esquema 1. La primera representación incluye compañías editoras que no han publicado ninguno de los libros que están actualmente en la biblioteca, mientras que el segundo caso excluye estas compañías editoras. Se elige la primera alternativa porque es menos restrictiva.

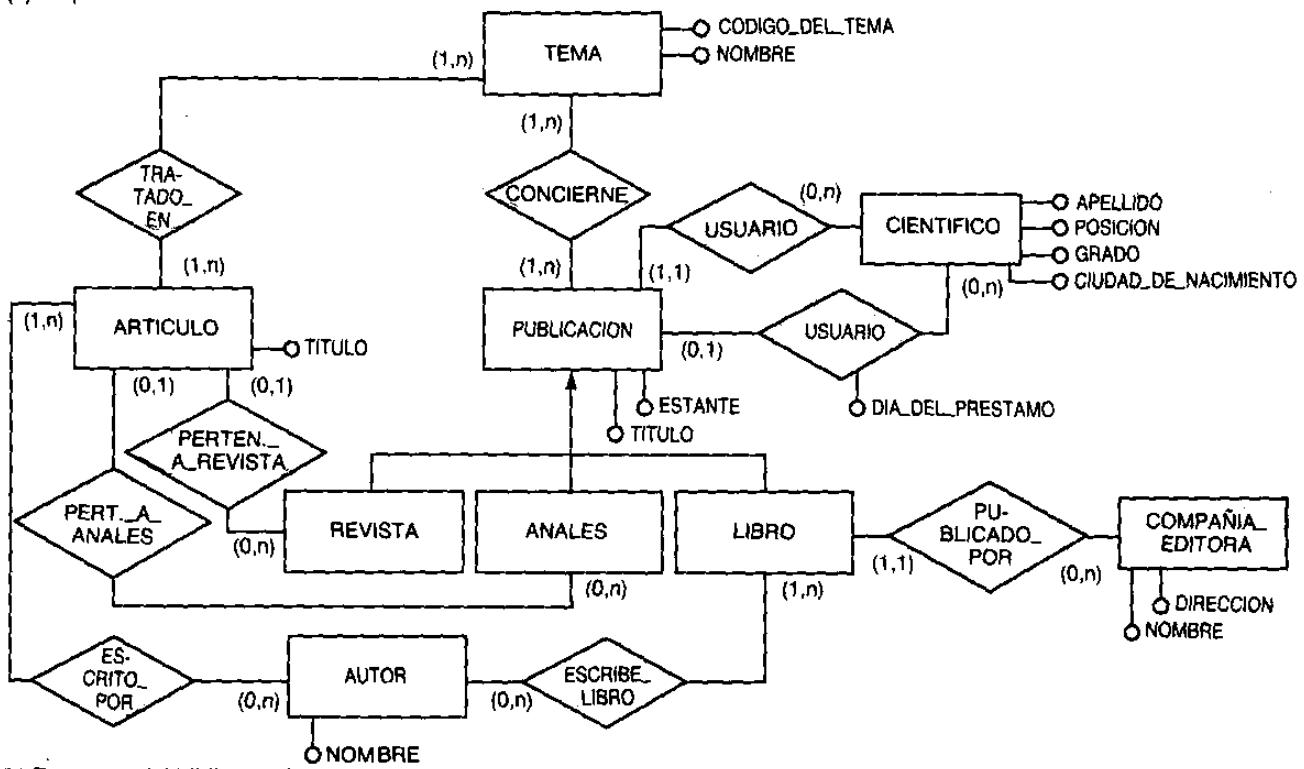
La segunda incompatibilidad se refiere al concepto AUTOR. El esquema 1 incluye a cualquier autor (como entidad), mientras haya un científico que se interese por sus publicaciones. El esquema 2 incluye sólo los autores (como atributos de ARTICULO) que tienen al menos un artículo disponible actualmente en la biblioteca. De nuevo, se elige la primera alternativa porque es menos restrictiva. Los productos finales del análisis de conflictos son los dos esquemas modificados de la figura 5.7.

5.4. Fusión de vistas

La *fusión de vistas* actúa sobre dos esquemas de entrada y produce un esquema que incluye todos los conceptos representados en los esquemas de entrada. En esta etapa del diseño se han resuelto todos los conflictos, así que la fusión de esquemas es una simple superposición de conceptos comunes. Las entidades que coinciden por completo se superponen de forma directa (por ejemplo, COMPAÑIA_EDITORA). Las entidades que corresponden a los mismos objetos de la realidad, pero que poseen diferentes atributos, se superponen tomando la unión de sus atributos; todos los identificadores se exponen en el esquema resultante.



(a) Esquema del científico



(b) Esquema del bibliotecario

Figura 5.7. Esquemas modificados.

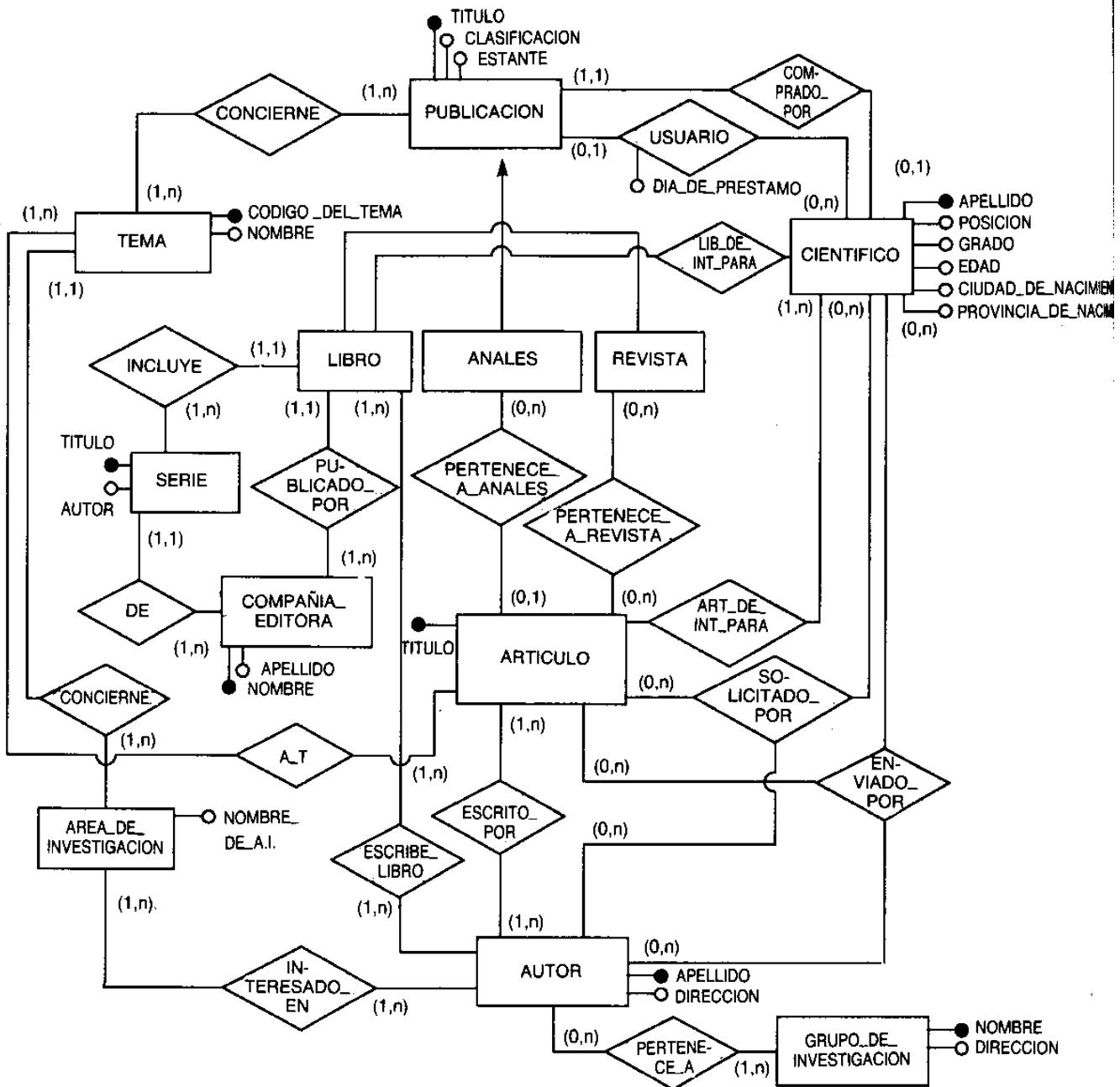


Figura 5.8. Esquema global después de la fusión.

Si las jerarquías de generalización se presentan como propiedades interesquemáticas, se añaden en este punto. Cuando las entidades están completamente superpuestas, todas las interrelaciones y generalizaciones de los esquemas de entrada figuran en el esquema resultante.

Las propiedades interesquemáticas también pueden conducir a reestructu-

raciones, adiciones o supresiones de conceptos en el esquema resultante. Por ejemplo, considérense las dos entidades TEMA y AREA.DE.INVESTIGACION. En la aplicación del bibliotecario, las áreas de investigación pueden dividirse en temas, que representan una subdivisión más fina de las disciplinas. Así, se puede añadir un enlace lógico entre las entidades AREA.DE.INVESTIGACION y TEMA, introduciendo una interrelación de uno-a-muchos entre ellos. El esquema global del ejemplo generado por la actividad de fusión se muestra en la figura 5.8.

Al final de la fusión de vistas se está en condiciones de evaluar la calidad del esquema integrado y, entonces, tratar de mejorarlo efectuando operaciones de reestructuración. Esta actividad se describe ampliamente en el capítulo 6.

5.5. Resumen

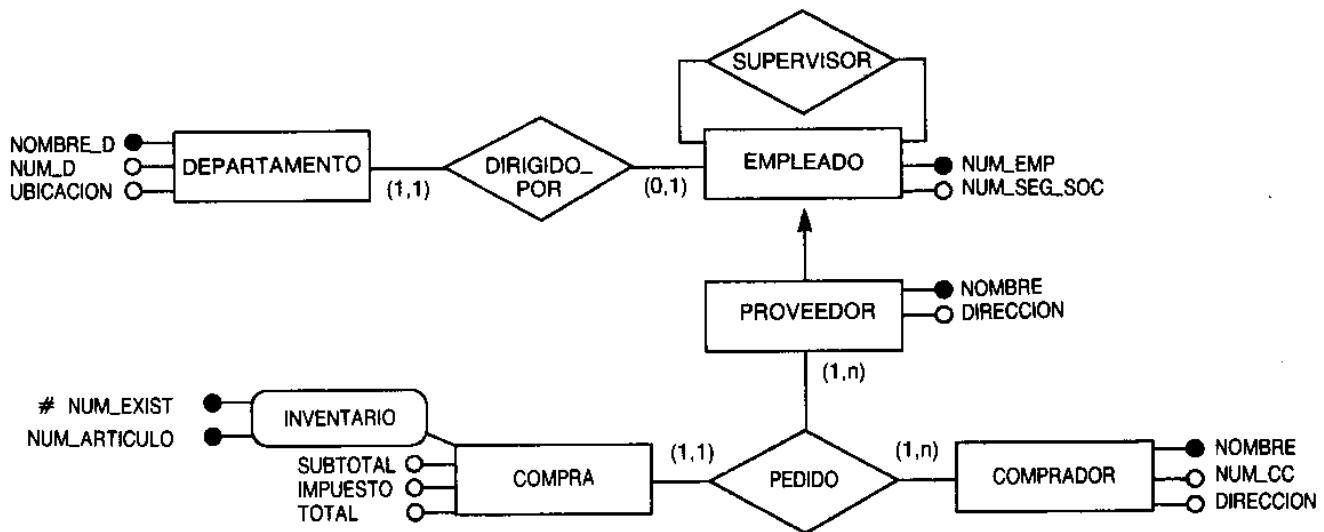
La integración de vistas es una actividad frecuente en el diseño de bases de datos. En este capítulo se ha expuesto una metodología para la integración de vistas. A gran escala, se sugiere lograr la integración de varios esquemas efectuando una secuencia de integraciones de dos esquemas a la vez; se toma como punto de partida el esquema armazón o la perspectiva *de gestión*, y los conceptos se agregan a él progresivamente. A pequeña escala, se sugiere que cada paso de integración se efectúe reconociendo primero los conflictos y luego resolviéndolos. Se presenta una posible alternativa para efectuar la búsqueda y resolución de los conflictos entre vistas. Los conflictos se originan en la diferencia de puntos de vista o en errores al nombrar o estructurar los datos; se pueden resolver por cambio de nombres o por reorganización local de los datos. Se mantiene la información acerca de los conflictos como propiedades interesquemáticas. Después de la resolución de conflictos, el paso final de fusión de los dos esquemas es relativamente fácil.

Ejercicios

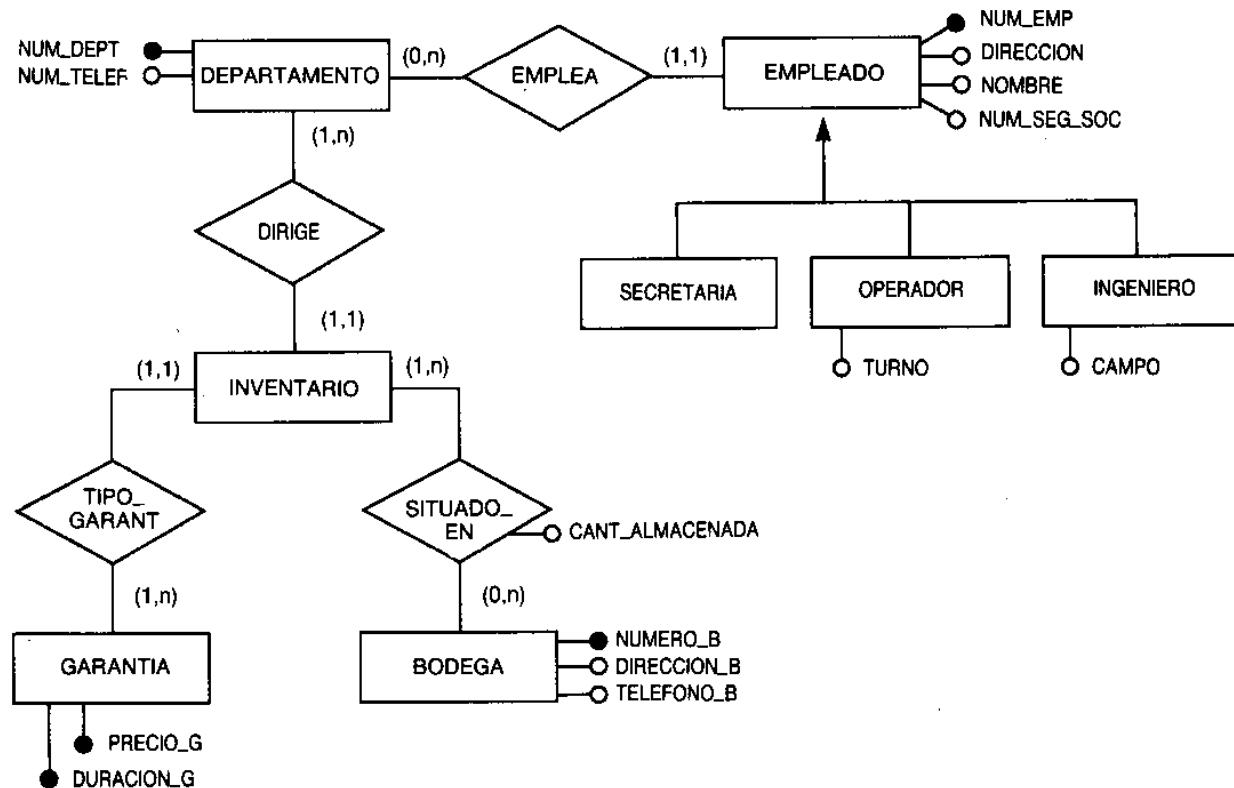
5.1. Invente varios ejemplos de lo siguiente:

- a) Representaciones equivalentes permitidas en el modelo de entidades-interrelaciones para un mismo concepto de la realidad.
- b) Perspectivas diferentes para los mismos conceptos en esquemas diferentes.
- c) Especificaciones de diseño incompatibles que den lugar a representaciones diferentes de las mismas propiedades.

5.2. Efectúe este experimento con uno de sus compañeros de clase o colegas. Identifique un dominio de aplicación conocido para ambos (por ejemplo, la estructura del centro de cálculo, del equipo de fútbol, de un restaurante,

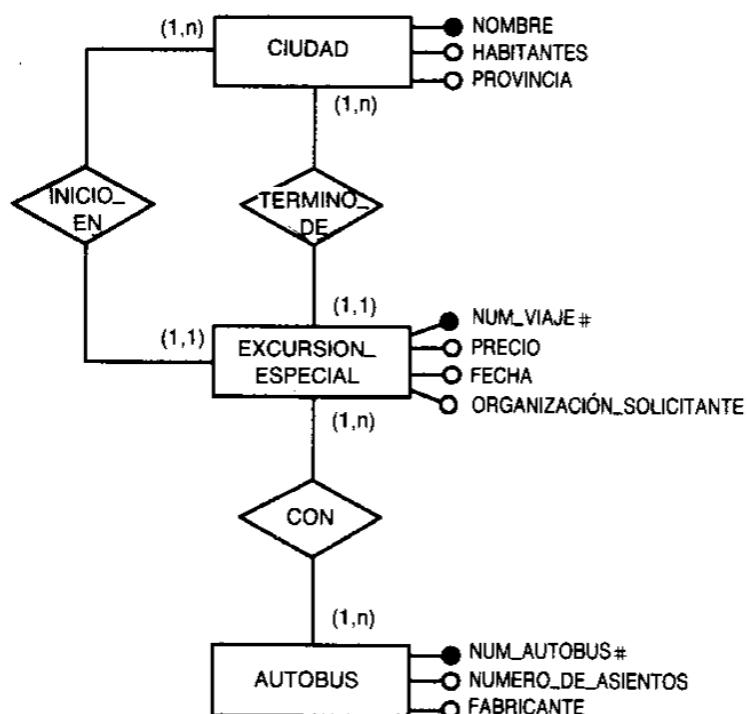


(a) Primer esquema

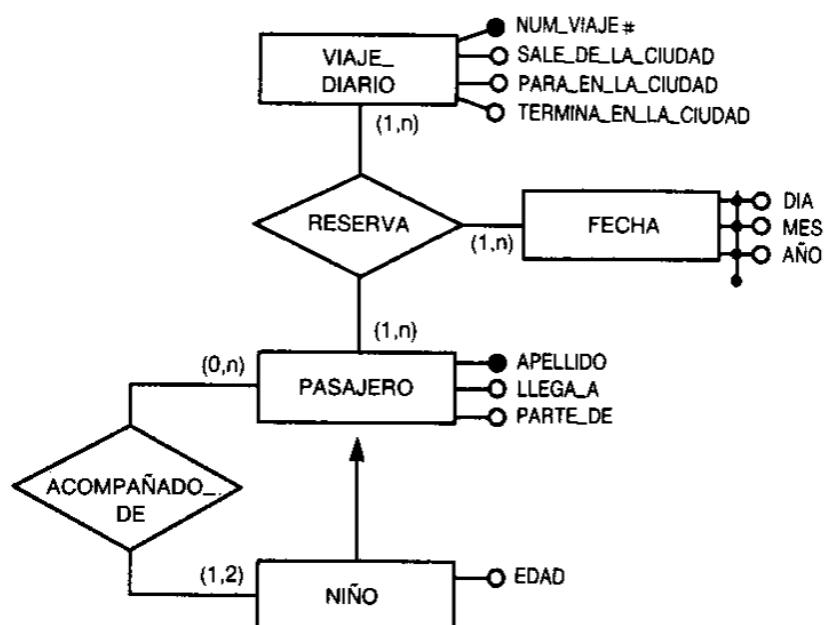


(b) Base de datos de una compañía

Figura 5.9. Base de datos de una compañía.

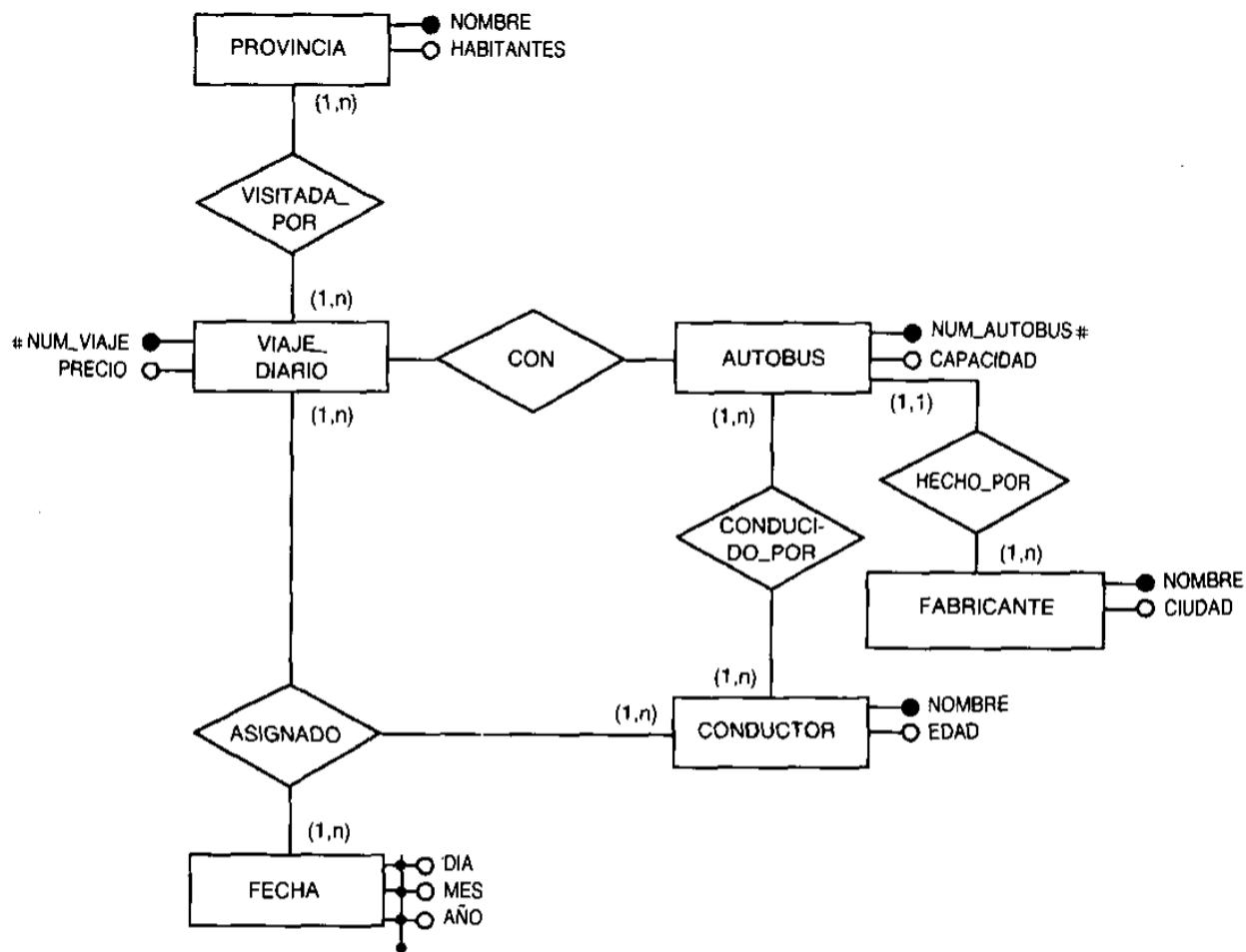


(a) Primer esquema



(b) Segundo esquema

Figura 5.10. Base de datos de viajes.



(c) Tercer esquema

Figura 5.10 bis. Base de datos de viajes (*continuación*).

etc.). Comenten, juntos, las características del dominio de aplicación, pero no muy a fondo. Luego, produzcan dos modelos conceptuales. Por último, comparén sus esquemas para ver si incluyen lo siguiente:

- Partes modeladas exactamente de la misma forma.
- Partes modeladas con construcciones equivalentes.
- Partes no comunes y propiedades interesquemáticas existentes entre ellas.
- Sinónimos y homónimos.
- Errores.

Luego, fusionen los esquemas e indiquen gráficamente lo siguiente:

- Los subesquemas que contienen todos los conceptos presentes en los dos esquemas de entrada.

- g) Los subesquemas que contienen todos los conceptos presentes en sólo un esquema.
- 5.3. Integre los dos esquemas de la figura 5.9, que representan ventas en una compañía y la estructura de sus departamentos y personal, produciendo un esquema único.
- 5.4. Integre los tres esquemas de la figura 5.10, que representan excursiones especiales, viajes diarios y reservas para viajes diarios, produciendo un esquema único.
- 5.5. Una vez creado un esquema global como resultado de una actividad de integración, conviene recordar la relación entre las construcciones ER de los esquemas de entrada y aquéllas de los esquemas globales. Defina un lenguaje que permita expresar esas correspondencias.
- 5.6. La metodología expuesta en este capítulo crea una única representación como resultado del proceso de integración. Defina una metodología que permita mantener varios puntos de vista diferentes de los usuarios respecto a la misma realidad, haciendo menos estrictas las reglas de fusión de conceptos.

Bibliografía

C. Batini, M. Lenzerini y S. Navathe, «A Comparative Analysis of Methodologies for Database Schema Integration», *ACM Computing Surveys*, 18, núm 4, diciembre de 1986, 323-64.

Este artículo presenta un estudio detallado de las metodologías para la integración de esquemas de bases de datos. El análisis abarca la integración de vistas y la integración de bases de datos, que se ocupa del problema de producir, en un entorno distribuido, un esquema integrado a partir de esquemas locales de bases de datos existentes. La primera parte del artículo analiza el papel del proceso de integración en bases de datos, sistemas de oficina y sistemas basados en conocimientos, y luego examina la influencia del modelo conceptual en el proceso de integración. La parte principal del artículo compara las metodologías de acuerdo con los siguientes criterios: el orden de las construcciones de los esquemas por integrar, las entradas y salidas consideradas, la estrategia del proceso de integración, los conflictos considerados y la procedimentalidad del proceso de integración. Se exponen las orientaciones de investigaciones futuras.

C. Batini y M. Lenzerini, «A Methodology for Data Schema Integration in the Entity-Relationship Model», *IEEE Transactions on Software Engineering*, SE-10, núm. 6, noviembre de 1984.

R. Elmasri y G. Wiederhold, «Data Model Integration Using the Structural Model». En P. Berstein, ed., *Proc. ACM-SIGMOD International Conference*, Boston, 1979.

S. B. Navathe y S. G. Gadgil, «A Methodology for View Integration in Logical Data Base Design». En *Proc. Eighth International Conference on Very Large Data Bases*, Ciudad de México, 1982.

Estas publicaciones describen tres enfoques hacia la integración de vistas que hacen hincapié en diferentes aspectos del proceso de integración. El primer artículo sugiere varios indicadores para orientar al diseñador en la investigación de los conflictos. Para cada conflicto se proponen varios guiones, es decir, soluciones típicas del conflicto. Se sugiere, asimismo, una actividad específica para mejorar la legibilidad del esquema global. El segundo artículo ofrece una taxonomía de tipos de conflictos y de operaciones de integración de esquemas. Se considera también el problema de automatizar el proceso de integración de vistas, distinguiendo las actividades que se pueden resolver automáticamente de las que necesitan una interacción con el diseñador o usuario. El tercer artículo considera el modelo de datos estructural, que posee una rica variedad de construcciones de modelado. Esta diversidad de construcciones da lugar a un extenso grupo de conflictos que se presentan y analizan.

- A. Motro y P. Buneman, «Constructing Superviews». En Y. Lien, ed., *Proc. ACM-SIGMOD International Conference*, Ann Arbor, Mich., 1981.

Este artículo presenta una metodología para la integración de esquemas de datos. La característica principal de la metodología es que ofrece un conjunto grande y poderoso de primitivas de reestructuración. Se consideran, asimismo, las especificaciones de procesamiento y las consultas.

- J. Biskup y B. Convent, «A Formal View Integration Method». En C. Zaniolo, ed., *Proc. ACM-SIGMOD International Conference*, Washington, D.C., 1986.

Se expone la integración de vistas en el marco de un modelo formal, que sirve de base teórica para una clara definición de varias operaciones implicadas en el proceso de integración (por ejemplo, comparaciones de contenido de información, transformaciones locales de esquemas y fusión de esquemas).

- R. Elmasri, J. Larson y S. B. Navathe, «Schema Integration Algorithms for Federated Databases and Logical Database Design», Honeywell Computer Sciences Center, informe técnico CSC-86-9:8212, 1986.

Este artículo es una descripción completa de una metodología y una herramienta para la integración n-aria, basada en el modelo entidad-categoría-interrelación. Ya se ha llevado a la práctica un prototipo. Incorpora una variedad de aserciones entre atributos y entre entidades para indicar las correspondencias.

- M. Mannino, S. B. Navathe y W. Effelberg, «A Rule-Based Approach for Merging Generalization Hierarchies», *Information Systems* 13, núm. 3, 1988, 257-72.

Este artículo trata un problema especial: la fusión de jerarquías de generalización de entidades de diferentes vistas. Se presenta una clasificación de todos los casos posibles, junto con un conjunto de reglas para la fusión de un par de jerarquías en cada caso.

- F. N. Civelek, A. Dogac y S. Spaccapietra, «An Expert-System Approach to View Definition and Integration». En C. Batini, ed., *Entity-Relationship Approach: A Bridge to the User: Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

Este artículo presenta un sistema experto que ayuda al usuario en varias actividades implicadas en el proceso de integración. El sistema ofrece apoyo a varias actividades de integración, un recurso de ayuda para usuarios principiantes y una explicación de los resultados logrados por el sistema.

- A. Savasere, A. Sheth, S.K. Gala, S.B. Navathe y H. Marcus, «On Applying Classification to Schema Integration». En *Proc. First International Workshop on Interoperability in Multi-Database Systems*, Kyoto, Japón, IEEE, abril de 1991.

Este artículo reciente describe un enfoque hacia la integración de esquemas que combina un modelo ER extendido como sección frontal con un modelo formal interno, CANDIDE, para la integración de esquemas. Se utiliza una variante del modelo ER empleado en este libro para introducir los criterios del usuario en la herramienta de integración de esquemas. Estas vistas o criterios se hacen corresponder a esquemas internos CANDIDE. En CANDIDE, se usa el concepto de clasificación para colocar las clases de diversos esquemas en un único grafo acíclico de clases. El artículo describe un conjunto de operaciones utilizadas en conjunción con el clasificador para lograr la integración.

Cómo mejorar la calidad de un esquema de base de datos

Igual que otros productos industriales, el esquema de una base de datos debe validarse antes de convertirse en un producto estable del diseño. Este proceso de validación se realiza revisando varias cualidades del esquema que se examinan en este capítulo: compleción, corrección, expresividad, legibilidad, minimalidad, autoexplicación, extensibilidad y normalidad. Estas cualidades deben revisarse periódicamente durante el proceso de diseño, y en realidad deben guiar dicho proceso. La prueba de estas cualidades es más eficaz cuando el diseño está completo, porque se tiene una visión global del esquema conceptual.

Las herramientas fundamentales disponibles para mejorar las cualidades de un esquema son las *transformaciones de esquemas*. Se ha visto, en el capítulo 3, cómo las transformaciones primitivas se pueden utilizar en el proceso de diseño; ahí, su papel es enriquecer el esquema con información nueva. Se han expuesto otras transformaciones en el capítulo 5, que se usan para alcanzar compatibilidad entre esquemas. En este capítulo, el objetivo de la aplicación de las transformaciones es *reestructurar* el esquema para producir una versión mejor, en términos de las cualidades recién mencionadas.

El capítulo se organiza como sigue. En el apartado 6.1 se definen formalmente las cualidades de un esquema de base de datos. El apartado 6.2 describe los tipos de transformaciones que pueden usarse para mejorar la calidad global del esquema. Los apartados 6.3, 6.4 y 6.5 tratan sobre cualidades específicas (minimalidad, autoexplicación, expresividad, legibilidad, normalidad) y definen varias reglas y métodos para lograrlas. El apartado 6.6 expone un ejemplo de reestructuración de esquemas: se utilizará el esquema final (Fig. 5.8) resultante de la actividad de integración del capítulo 5.

6.1. Cualidades de un esquema de base de datos

En este apartado se da una respuesta a la pregunta: ¿qué es un buen esquema? Se distinguen varias cualidades típicas de un esquema conceptual.

Compleción. Un esquema es completo cuando representa todas las características pertinentes del dominio de aplicación. La compleción puede, en principio, comprobarse: 1) mirando con detalle todos los requerimientos del dominio de aplicación y verificando que cada uno de ellos esté representado en algún lugar del esquema (en este caso, se dice que *el esquema es completo respecto a los requerimientos*), y 2) revisando el esquema para ver que cada concepto se mencione en los requerimientos (en este caso, se dice que *los requerimientos están completos respecto al esquema*).

La compleción puede comprobarse más eficazmente comparando el esquema de datos con el esquema de funciones. Se tratará este aspecto en el capítulo 9, que se refiere al análisis conjunto de datos y funciones. Por esta razón, no se insiste más en la compleción en este capítulo.

Corrección. Un esquema es correcto cuando usa con propiedad los conceptos del modelo ER. Se puede distinguir dos tipos de corrección, sintáctica y semántica. Un esquema es sintácticamente correcto cuando los conceptos se definen con propiedad en el esquema; por ejemplo, los subconjuntos y las generalizaciones se definen entre entidades pero no entre interrelaciones. Un esquema es semánticamente correcto cuando los conceptos (entidades, interrelaciones, etc.) se usan de acuerdo con sus definiciones. Por ejemplo, es un error semántico usar un atributo para representar los productos en la base de datos de una empresa manufacturera cuando se necesita representar varias propiedades de los productos (por ejemplo, código-de-producto, precio, partes, etc.), porque un atributo es una propiedad elemental. La siguiente lista muestra los errores semánticos más frecuentes:

1. Usar un atributo en lugar de una entidad
2. Olvidar una generalización (o un subconjunto)
3. Olvidar la propiedad de herencia de las generalizaciones
4. Usar una interrelación con un número erróneo de entidades (por ejemplo, una interrelación binaria en lugar de una ternaria)
5. Usar una entidad en lugar de una interrelación
6. Olvidar algún identificador de una entidad, especialmente identificadores compuestos externos
7. Omitir alguna especificación de cardinalidad mínima o máxima

Minimalidad. Un esquema es **mínimo** cuando cada aspecto de los requerimientos aparece sólo una vez en el esquema. También se puede decir que un esquema es mínimo si no se puede borrar del esquema un concepto sin perder

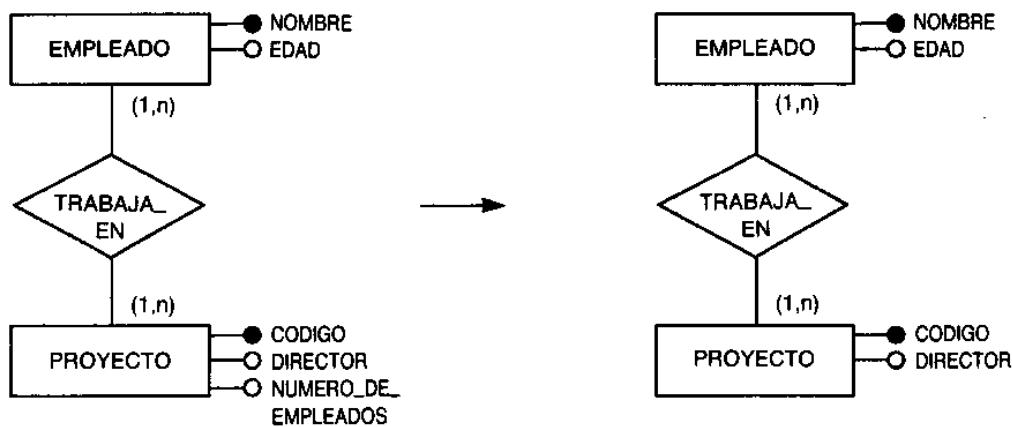


Figura 6.1. Un esquema redundante.

alguna información. El esquema de la figura 6.1 representa empleados y los proyectos en los que trabajan. Uno de los atributos de la entidad PROYECTO es NUMERO_DE_EMPLEADOS, que se puede derivar del esquema simplemente contando los empleados relacionados con el proyecto. Por tanto, el esquema no es mínimo y el atributo NUMERO_DE_EMPLEADOS se puede borrar sin cambiar el contenido de información del esquema. Cabe señalar que algunas veces es aconsejable permitir alguna redundancia en el esquema; sin embargo, esta redundancia debe documentarse. Esto se logra, por lo regular, añadiendo al esquema conceptual una tabla que indica cómo se calculan los datos derivados a partir de otros datos. La minimalidad se tratará en el apartado 6.2.

Expresividad. Un esquema es **expresivo** cuando representa los requerimientos de una forma natural y se puede entender con facilidad a través del significado de las construcciones de los esquemas ER, sin necesidad de explicaciones adicionales. Como ejemplo de expresividad, considérese el esquema de la figura 6.2, el cual describe la enseñanza y calificación de seminarios y cursos. La expresividad mejora al introducir las nuevas entidades PERSONAL_DE_ENSEÑANZA (una generalización de las entidades PROFESOR e INSTRUCTOR) y ofertas (generalización de las entidades CURSO y SEMINARIO) y relacionándolas mediante la única interrelación ENSEÑA.

Legibilidad. Esta es una propiedad del diagrama que representa gráficamente al esquema. Un diagrama tiene buena legibilidad cuando respeta ciertos criterios estéticos que hacen el diagrama elegante. Los principales criterios son los siguientes:

1. Los diagramas deben trazarse en hojas cuadriculadas, para que los cuadros que representan entidades y los rombos que indican interrelaciones tengan

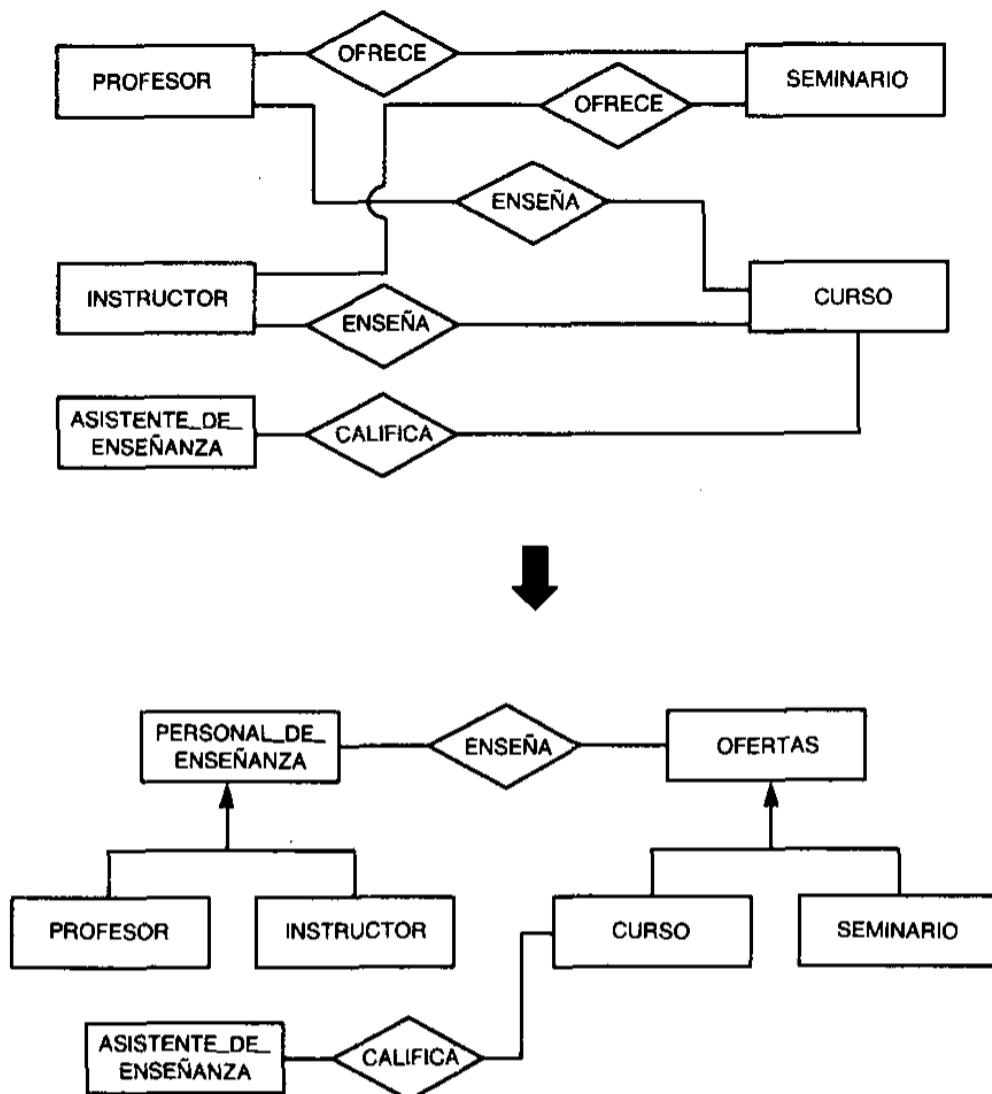
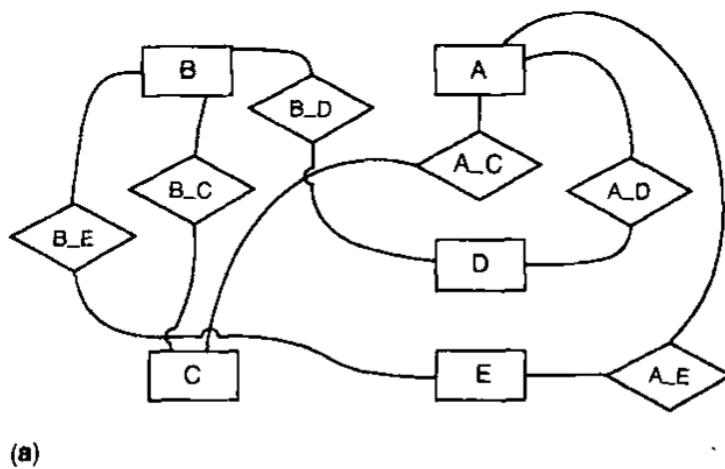


Figura 6.2. Mejoramiento de la expresividad.

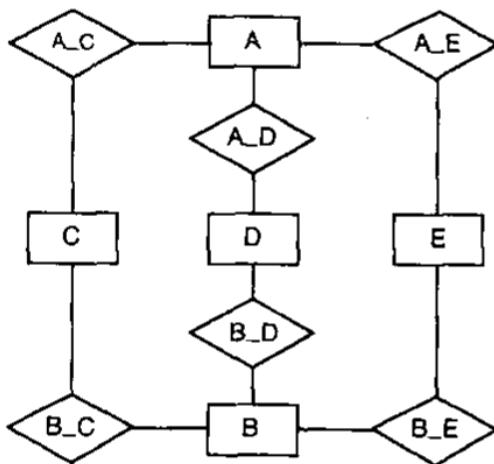
aproximadamente el mismo tamaño y las conexiones sean trazos horizontales y verticales.

2. Se debe destacar las estructuras simétricas.
3. Se minimiza el número global de cruces (los cruces frecuentes disminuyen el *ancho de banda de percepción* del lector).
4. Debe minimizarse el número global de esquinas a lo largo de las conexiones.
5. En las jerarquías de generalización, la entidad padre debe situarse arriba de las entidades hijo, y los hijos deben situarse simétricamente respecto al padre. De forma similar, en las interrelaciones de subconjunto, la entidad padre debe situarse arriba y la entidad subconjunto abajo.

Como demostración de legibilidad, considérese el esquema de la figura 6.3.



(a)

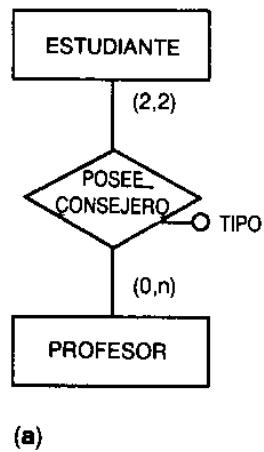


(b)

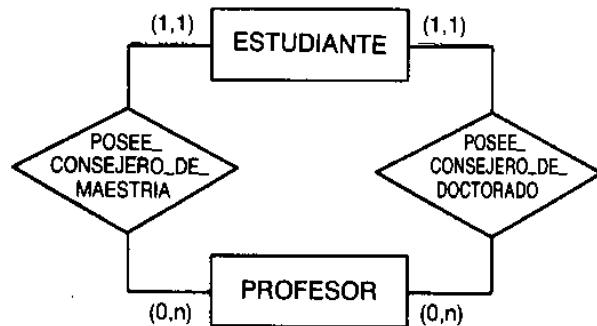
Figura 6.3. Mejoramiento de la legibilidad.

La legibilidad mejora en la figura 6.3b, al omitir los cruces y destacar la simetría.

Autoexplicación. Un esquema se explica a sí mismo cuando puede representarse un gran número de propiedades usando el modelo conceptual por sí mismo, sin otros formalismos (por ejemplo, anotaciones en lenguaje natural). Como ilustración de un esquema que no es autoexplicativo, considérese la representación de estudiantes y sus consejeros de maestría (máster) y doctorado. Supóngase que cada estudiante posee como máximo un consejero de maestría y uno de doctorado, y que el mismo estudiante puede (en momentos distintos) ser tanto estudiante de maestría (máster) como de doctorado. Esta restricción no puede representarse por completo en el esquema de la figura 6.4a, porque



(a)



(b)

Figura 6.4. Ejemplo de interrelaciones autoexplicativas.

ningún concepto del modelo permite declarar que «si un objeto estudiante pertenece a dos casos de la interrelación POSEE_UN_CONSEJERO, el atributo TIPO debe adoptar dos valores distintos». Si, en cambio, se usan dos interrelaciones diferentes entre estudiantes y profesores (Fig.6.4b), se podrá imponer la restricción definiendo los valores adecuados de las cardinalidades mínima y máxima de las interrelaciones. La expresividad y la autoexplicación se tratan en el apartado 6.3.

Extensibilidad. Un esquema se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes (módulos o vistas), a fin de aplicar los cambios dentro de cada parte. Se tratará la extensibilidad en el capítulo 7, donde se definen criterios para la modularización y la documentación descendente de esquemas y se usan esos conceptos para el mantenimiento de esquemas.

Normalidad. El concepto de normalidad viene de la teoría de la normalización, asociada al modelo relacional. Las *formas normales* (primera, segunda,

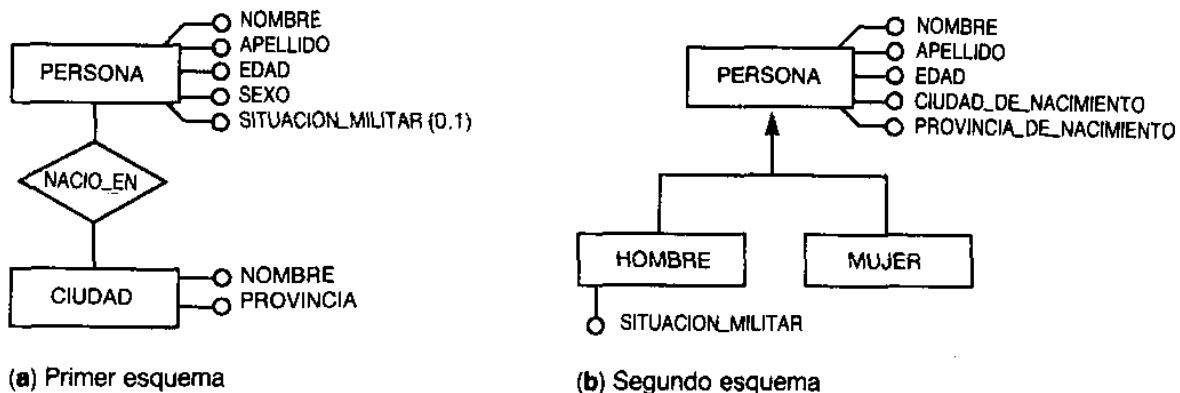


Figura 6.5. Dos esquemas con el mismo contenido de información.

tercera, cuarta, y una variación de la tercera forma normal llamada forma normal de *Boyce-Codd*), pretenden mantener la estructura lógica de los datos en una forma normal limpia, «purificada», mitigando los problemas de las anomalías de inserción, borrado y actualización que ocasionan trabajo innecesario porque deben aplicarse los mismos cambios a varios casos de datos, así como el problema de la pérdida accidental de datos o la dificultad de representación de determinados hechos. Si desea un tratamiento completo de las formas normales y la normalización, el lector deberá consultar los capítulos 13 y 14 del texto de Elmasri/Navathe (véase la bibliografía del capítulo 1).

La teoría de la normalización relacional utiliza los conceptos de clave y dependencia funcional. Se introducirán estos conceptos en el apartado 6.5, y luego se aplicarán en el modelo ER. Nótese que, implícitamente, se ha captado la mayor parte de la información de dependencias en el modelo ER, en términos de las cardinalidades mínima y máxima y de los identificadores.

6.2. Transformaciones de esquemas

Las transformaciones de esquemas se aplican a un esquema de entrada S_1 y producen un esquema resultante S_2 . Pueden clasificarse de acuerdo con el tipo de mejora de calidad que producen. Inicialmente, las transformaciones se analizan en términos de cómo cambian el *contenido de información* de un esquema. Es muy difícil definir formalmente el contenido de información del esquema o probar que el contenido de información de dos esquemas es idéntico. Una forma de comparar el contenido de información de dos esquemas S_1 y S_2 es comparar su capacidad para responder a consultas: se dice que S_1 y S_2 poseen el mismo contenido de información (o que son *equivalentes*) si, por cada consulta Q que pueda expresarse sobre S_1 , existe una consulta Q' que puede expresarse sobre S_2 dando la misma respuesta, y viceversa.

Por ejemplo, los dos esquemas de la figura 6.5 tienen el mismo contenido de información, pues por cada consulta que puede expresarse sobre PERSONA y CIUDAD en el primer esquema, existe una pregunta correspondiente en el segundo esquema que proporciona la misma respuesta (como puede comprobarse intuitivamente). De forma similar, se dice que el contenido de información de un esquema A es *mayor* que el de un esquema B si existe alguna consulta Q sobre A que no tenga una consulta correspondiente sobre B, pero no viceversa.

Ahora estamos en condiciones de clasificar las transformaciones como sigue:

- 1. Transformaciones que preservan la información:** el contenido de información del esquema no es alterado por la transformación.
- 2. Transformaciones que cambian la información,** clasificadas más en detalle como sigue:
 - a) Transformaciones de aumento:** el contenido de información del esquema resultante es mayor que el del esquema de entrada.
 - b) Transformaciones de reducción:** el contenido de información del esquema resultante es menor que el del esquema de entrada.
 - c) Transformaciones no comparables,** en cualquier otro caso.

Hasta ahora hemos tratado principalmente con transformaciones de aumento o no comparables, aplicadas durante el proceso de diseño e integración. Por ejemplo, cuando se introduce una nueva propiedad interesquemática en un esquema integrado, aumenta el contenido de información del mismo; cuando se cambia un nombre por otro para eliminar un homónimo, se produce un concepto nuevo, no comparable.

La comprobación de la calidad hace uso frecuente de las transformaciones que preservan la información: el contenido de información del esquema conceptual no debe variar, pero la organización de los conceptos debe mejorar. Las transformaciones de reducción sólo se usan cuando el esquema conceptual contiene conceptos superfluos que no están expresados en los requerimientos y, por tanto, deben eliminarse. La tabla 6.1 resume los tipos de transformaciones usadas más comúnmente durante el diseño conceptual.

6.3. Transformaciones para lograr minimalidad

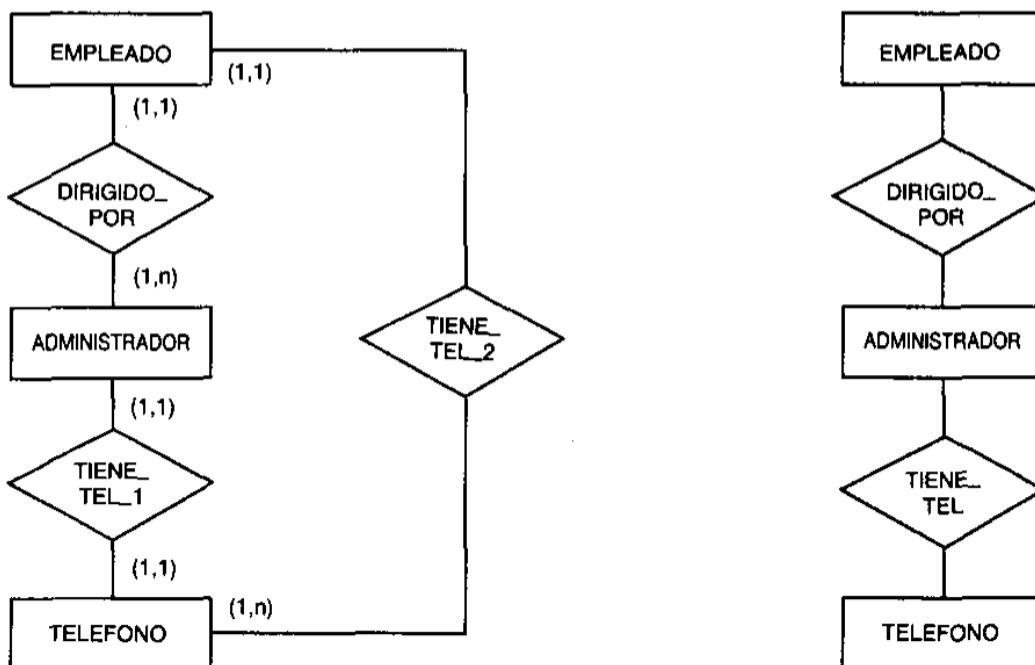
La redundancia en los esquemas ER puede surgir por varias razones. Una de las más comunes es que los requerimientos poseen una redundancia inherente, y ésta a veces se comunica al esquema. También, en una metodología ascendente, la redundancia puede surgir cuando se expresan en diferentes esquemas conceptos relacionados, y se hace evidente cuando los esquemas se fusionan. La redun-

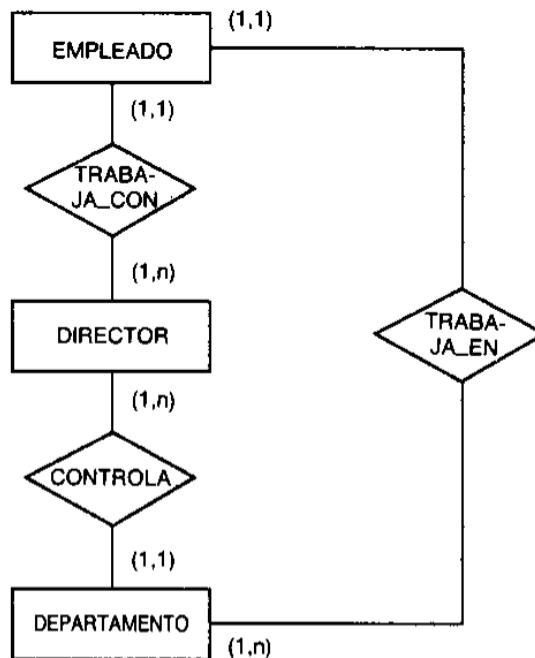
Tabla 6.1. Transformaciones de esquemas usadas durante el diseño conceptual.

Tipo de transformación	Usadas en
Aumento	Diseño descendente (capítulos 3-4)
Aumento	Diseño ascendente (capítulos 3-4)
No comparable	Resolución de conflictos (capítulo 5)
Aumento	Aumento de propiedades interesquemáticas (capítulo 5)
Preservación	Reestructuración para lograr minimalidad, legibilidad y normalidad (capítulo 6)

dancia en esquemas conceptuales ER está inmersa en los siguientes aspectos de los esquemas.

Ciclos de interrelaciones. La redundancia se da cuando una interrelación R_1 entre dos entidades posee el mismo contenido de información que una ruta de interrelaciones R_2, R_3, \dots, R_n que conecta exactamente los mismos pares de casos de entidades que R_1 . Es claro que no todos los ciclos de interrelaciones son fuentes de redundancia; como ejemplo, considérese el esquema de la figura 6.6, que representa empleados, administradores y teléfonos. Si la interrelación `POSEE_TEL_2` conecta a cada empleado con el teléfono o teléfonos de su admi-

**Figura 6.6.** La minimalidad depende del significado.



La interrelación TRABAJA_CON es equivalente a la ruta TRABAJA_EN, CONTROLA.
 La interrelación CONTROLA es equivalente a la ruta TRABAJA_CON, TRABAJA_EN.

Figura 6.7. Ejemplo de ciclo redundante de interrelaciones.

nistrador, el esquema es redundante; de lo contrario, el esquema es mínimo. De forma más general, los ciclos de entidades e interrelaciones en un esquema pueden o no introducir redundancia, dependiendo de su significado.

Considérese ahora el esquema de la figura 6.7. El esquema representa datos acerca de los empleados, directores y departamentos; un director puede controlar varios departamentos, pero cada departamento tiene solamente un director. Las interrelaciones TRABAJA_CON y CONTROLA son mutuamente redundantes, mientras que la interrelación TRABAJA_EN no lo es. Cuando un ciclo contiene varias interrelaciones mutuamente redundantes, es posible eliminar cualquiera de ellas.

A pesar de que la redundancia en los ciclos de interrelaciones depende del significado, existen obvias comprobaciones sintácticas que pueden realizarse para detectar la redundancia. Por ejemplo, la interrelación que se obtiene al combinar dos o más interrelaciones de uno a uno es, asimismo, una interrelación de uno a uno. Por tanto, no puede ser equivalente a una interrelación de uno a muchos, de muchos a uno o de muchos a muchos. De manera similar, la interrelación obtenida al combinar dos o más interrelaciones de uno a muchos es también de uno a muchos; de modo que no puede ser equivalente a una interrelación de uno a uno, de muchos a uno o de muchos a muchos. En todos los demás casos, la cardinalidad de la interrelación combinada no se puede deducir;

por ejemplo, la combinación de una interrelación de uno a muchos con una interrelación de muchos a uno puede producir cualquier tipo de interrelación (puede ser de uno a uno, uno a muchos, muchos a uno o muchos a muchos).

Además de la equivalencia, los ciclos pueden generar algún tipo de *restricciones de contención*. Por ejemplo, supóngase que ESTUDIANTE y CURSO están relacionados por la interrelación binaria MATRICULADO_EN y EXAMINADO; entonces, a pesar de que las interrelaciones no son equivalentes (y por tanto el ciclo no es redundante), se tiene la clara restricción de que un estudiante no puede ser examinado en un curso en el que no esté matriculado. Por ende, la interrelación EXAMINADO está contenida dentro de la interrelación MATRICULADO_EN. Este tipo particular de redundancia puede evitarse incluyendo un atributo booleano EXAMEN_TOMADO en la interrelación MATRICULADO_EN y eliminando la interrelación EXAMINADO.

Atributos derivados. La redundancia puede deberse a la existencia de un algoritmo para calcular los valores de datos derivados de los otros datos; de aquí que los datos derivados puedan omitirse de un esquema ER mínimo. Ya vimos varios ejemplos de datos derivados en el capítulo 4, en el ejemplo del impuesto sobre la renta. Los datos derivados pueden ser extremadamente útiles para mejorar la eficiencia de una base de datos; este criterio decidirá, en última instancia, la conveniencia de almacenar datos derivados durante un diseño lógico. Se recomienda incluir los datos derivados redundantes en el esquema conceptual, pero indicando con claridad las reglas pertinentes para su cálculo.

Subconjuntos implícitos. Después de la integración de esquemas, algunos de los subconjuntos podrían derivarse de otros subconjuntos presentes en el esquema. Como ejemplo, considérense los dos esquemas de la figura 6.8: el subconjunto entre EMPLEADO y ANALISTA en el esquema después de la integración, es derivable; por tanto, se puede eliminar el subconjunto del esquema sin cambiar su contenido de información.

En resumen, es tarea del diseñador decidir si acepta la redundancia en el esquema conceptual o la elimina. En cualquier caso, la redundancia puede ser fuente de anomalías en la administración de datos; por ello, debe estar claramente indicada en el esquema. Se presenta un estudio sistemático sobre el tratamiento de las redundancias en el capítulo 11.

6.4. Transformaciones para lograr expresividad y autoexplicación

La autoexplicación se logra cuando las propiedades de los datos se expresan usando solamente el propio modelo conceptual, en vez de anotaciones adicionales; la expresividad se realza simplificando el esquema.

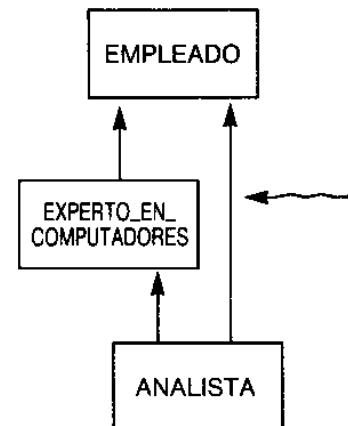
Se describen aquí algunas transformaciones típicas efectuadas para mejorar la expresividad.



(a) Primer esquema



(b) Segundo esquema



(c) Esquema integrado

Figura 6.8. Ejemplo de subconjunto implícito.

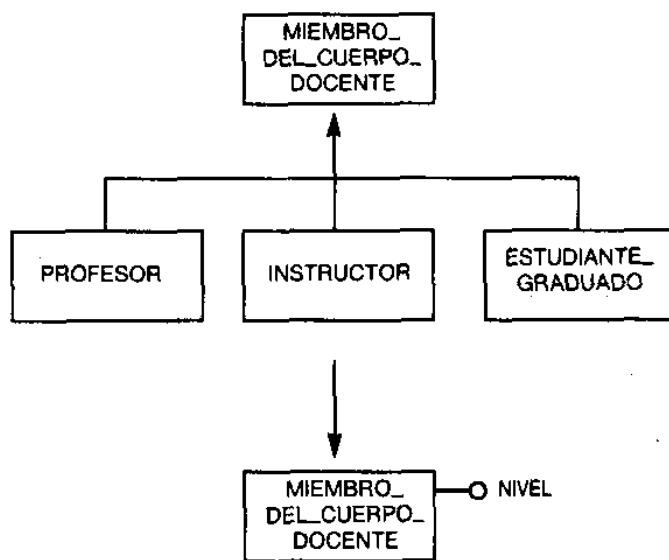


Figura 6.9. Eliminación de una jerarquía de generalización no representativa.

Eliminación de subentidades colgantes en las jerarquías de generalización. Puede suceder que el diseñador cree una generalización en el proceso de asignar diferentes propiedades a entidades de la jerarquía. Si, al final del proceso de diseño, las subentidades no se distinguen por ninguna propiedad específica, pueden reducirse a la superentidad. La diferencia entre las distintas entidades se expresa entonces mediante un atributo. Un ejemplo de eliminación de un subtipo se muestra en la figura 6.9; los valores del dominio del atributo NIVEL son (PROFESOR, INSTRUCTOR, ESTUDIANTE_GRADUADO) o cualquier esquema de codificación que incluya estas opciones.

Esta transformación se puede aplicar también cuando las subentidades tienen sólo unas cuantas propiedades distintivas; en este caso, la superentidad adquiere todas las propiedades de las subentidades colapsadas. Esta transformación produce un esquema más simple y genera una típica situación de conflicto entre la expresividad y la autoexplicación: un esquema compacto es a veces más comprensible que uno más grande; sin embargo, al colapsar entidades se puede perder precisión en la descripción de sus propiedades.

Eliminación de entidades colgantes. Considérese *colgante* una entidad E si posee pocos (posiblemente uno) atributos A_i y una conexión con otra entidad (la entidad *principal*) a través de una interrelación R; en este caso, puede ser conveniente simplificar el esquema eliminando la entidad colgante y la interrelación de conexión, pasando los atributos A_i de la entidad colgante a la entidad principal. Como en el caso anterior, las entidades colgantes pueden haberse generado durante el proceso de asignarles más propiedades. La figura 6.10 muestra un ejemplo de esta transformación: el concepto CIUDAD_DE_NACIMIENTO se

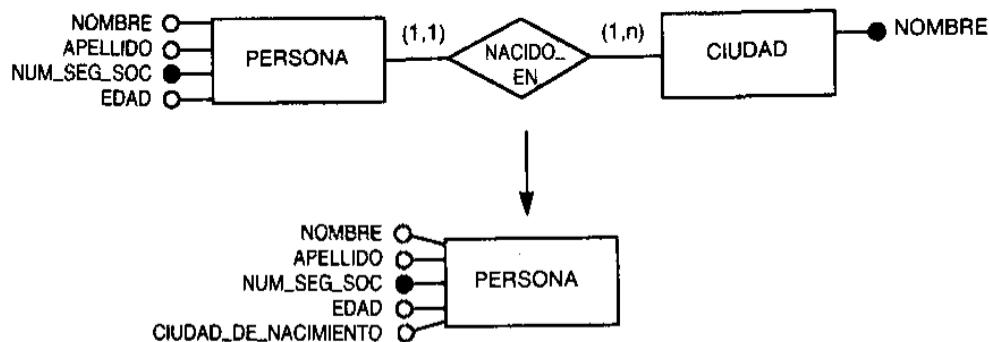


Figura 6.10. Eliminación de una entidad colgante.

ha representado temporalmente como una entidad, pero puede sintetizarse como un simple atributo de la entidad PERSONA.

Las cardinalidades mínima y máxima de un atributo A nuevo (respecto a la entidad principal) pueden calcularse fácilmente como una combinación de las cardinalidades anteriores de la entidad colgante E en la interrelación R y del atributo original A dentro de la entidad. Por ejemplo, si la cardinalidad máxima de E es 1 y la de A es n, la cardinalidad máxima del nuevo atributo es n.

Creación de una generalización. Esta transformación se aplica cuando se descubren dos entidades distintas con propiedades similares, que pertenecen en realidad a la misma jerarquía de generalización. Ya vimos un ejemplo de esta transformación en la figura 6.1. La adición de una generalización proporciona sencillez y concisión al esquema resultante mediante el uso de la propiedad de herencia.

Creación de un nuevo subconjunto. Esta transformación destaca el papel de una entidad. Se puede aplicar a entidades con cardinalidad mínima de cero en una interrelación; esto significa que la interrelación se aplica a un subconjunto de los casos de la entidad. Esta transformación debe aplicarse cada vez que se mejante subconjunto posea una identidad clara y sea significativo para el diseño. Por ejemplo, en la figura 6.11, la entidad EMPLEADO puede o no ser un conductor de vehículos. Esto puede modelarse introduciendo un subtipo especial de los empleados, llamado CONDUCTOR.

6.5. Transformaciones para lograr normalización

En el modelo relacional, la normalización de relaciones es un proceso de aplicación de transformaciones progresivas para lograr la forma normal deseada. El proceso está guiado por las *dependencias funcionales*. En este apartado se ana-

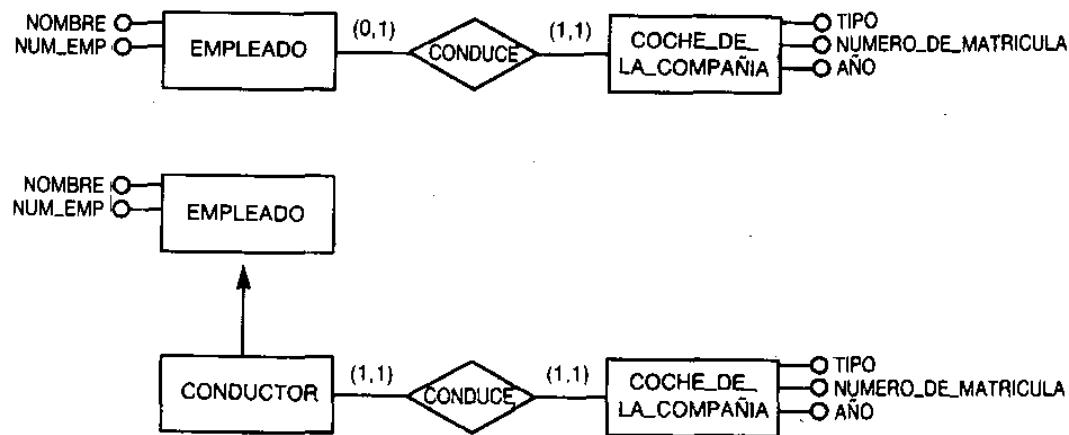


Figura 6.11. Creación de un nuevo subconjunto.

liza cómo los conceptos de dependencia funcional y formas normales pueden aplicarse al modelo ER.

6.5.1. Dependencias funcionales

Existe una **dependencia funcional** (DF) entre dos atributos monovalentes, A_1 y A_2 , de una entidad E o de una interrelación R, si cada valor de A_1 corresponde exactamente a un valor de A_2 . Esta propiedad se estudia mejor con ejemplos de casos. Sean A_1 , A_2 dos atributos de una entidad E; supóngase que existe un caso de entidad e_1 de E, que tiene los valores a_1 para A_1 y a_2 para A_2 :

$\langle e_1: \dots a_1 \dots a_2 \dots \rangle$

La dependencia funcional entre A_1 y A_2 implica que si existe otro caso de entidad, e_2 , en el que A_1 adopta el valor a_1 , entonces A_2 debe adoptar el valor a_2 :

$\langle e_2: \dots a_1 \dots a_2 \dots \rangle$

Decimos que A_1 **determina funcionalmente** a A_2 , lo cual se denota también como $A_1 \rightarrow A_2$; el atributo a la izquierda de la DF se llama el **determinante**. Las DF se establecen de forma similar entre conjuntos de atributos; por ejemplo, $A_1, A_2 \rightarrow A_3$ (el par de atributos $[A_1, A_2]$ es, en este caso, el determinante) o $A_1 \rightarrow A_2, A_3$. Cuando el lado derecho de una DF es un conjunto S de n atributos, la DF original es equivalente a n dependencias funcionales individuales, cada una con un atributo sencillo de S como lado derecho. Por ejemplo, la dependencia $A_1 \rightarrow A_2, A_3$ es equivalente a las dos DF $A_1 \rightarrow A_2$ y $A_1 \rightarrow A_3$ por separado. Las dependencias funcionales se establecen, asimismo, entre atributos de interrelaciones, exactamente con la misma interpretación.



Figura 6.12. Entidad PERSONA.

El lector cuidadoso debe haberse dado cuenta en este punto de que, *en un esquema correcto*, todos los identificadores internos de las entidades determinan funcionalmente a los otros atributos monovalentes. El ejemplo de la figura 6.12 muestra la entidad PERSONA con dos identificadores internos, NUM_SEG_SOC y el par (NOMBRE, FECHA_DE_NACIMIENTO). Otros atributos son DIRECCION, CIUDAD, PROVINCIA y CODIGO_POSTAL. Por tanto, tenemos las siguientes DF:

$\text{NUM_SEG_SOC} \rightarrow \text{NOMBRE, FECHA_DE_NACIMIENTO, DIRECCION, CIUDAD, PROVINCIA, CODIGO_POSTAL}$

$\text{NOMBRE, FECHA_DE_NACIMIENTO} \rightarrow \text{DIRECCION, CIUDAD, NUM_SEG_SOC, PROVINCIA, CODIGO_POSTAL}$

Estas dependencias dicen que si se asigna el valor de los atributos determinantes, se encontrará en la base de datos un valor de los atributos determinados. Esto es una consecuencia trivial del hecho de que los identificadores son también determinantes. Asimismo, se tiene una DF adicional,

$\text{CODIGO_POSTAL} \rightarrow \text{CIUDAD, PROVINCIA}$

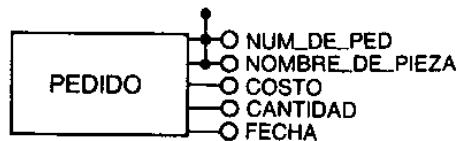
que expresa la propiedad de que las personas con el mismo CODIGO_POSTAL viven en la misma CIUDAD y PROVINCIA.

6.5.2. Anomalías de actualización

Si bien las dependencias funcionales que corresponden a identificadores no causan problemas, otras dependencias que pueden existir en una entidad pueden causar las llamadas *anomalías de actualización*, que se presentan de modo informal en este apartado. Considérese el ejemplo de la figura 6.13, que describe PEDIDO en términos de números de pedido, piezas pedidas, fecha de pedido, costo de cada pieza y cantidad pedida de cada pieza. La figura 6.13b muestra algunos casos de entidad.

El identificador de la entidad está dado por el par (NUM_PED, NOMBRE_DE_PIEZA); de aquí, se deduce la siguiente dependencia:

$\text{NUM_PED, NOMBRE_DE_PIEZA} \rightarrow \text{COSTO, CANTIDAD, FECHA}$



(a) La entidad PEDIDO

<01 : 1518; BOLIGRAFO, 1, 12, 3-8-90>
 <02 : 1518, LAPICERO, 0.5, 15, 3-8-90>
 <03 : 1521, LAPICERO, 0.5, 18, 2-9-89>
 <04 : 1407, BOLIGRAFO, 1, 15, 2-6-89>
 <05 : 1407, BORRADOR, 0.2, 28, 2-6-89>
 <06 : 1407, PIZARRA, 5, 3, 2-6-89>
 <07 : 1729, BORRADOR, 0.2, 1, 3-1-90>
 <08 : 1729, DISQUETE, 2, 10, 3-1-90>
 <09 : 1729, LAPICERO, 0.5, 15, 3-1-90>

(b) Casos de la entidad PEDIDO

Figura 6.13. Anomalías de actualización.

Sin embargo, el costo de una pieza está determinado de manera única por su número de pieza; esto se afirma mediante la siguiente dependencia funcional:

$\text{NOMBRE_DE_PIEZA} \rightarrow \text{COSTO}$

Esta es la causa de la redundancia mostrada en los casos de la entidad; por ejemplo, el costo de los lápices se repite tres veces. Más aún, se reconocen las siguientes anomalías:

1. **Anomalía de inserción:** No se puede indicar el costo de una pieza, a menos que tenga algunos pedidos pendientes.
2. **Anomalía de eliminación:** Cuando se borra el último pedido pendiente para una pieza, también se borra la información sobre su costo.
3. **Anomalía de actualización:** Si cambia el costo de algunas piezas, todos los pedidos referentes a esas piezas cambian también; la operación de actualización se propaga a varios casos de la entidad.

Todas estas anomalías se relacionan con la presencia de una dependencia funcional *indeseable*, a saber, $\text{NOMBRE_DE_PIEZA} \rightarrow \text{COSTO}$. Una anomalía semejante se debe a la dependencia $\text{NUM_PED} \rightarrow \text{FECHA}$. El proceso de normalización es una progresiva detección y eliminación de esas dependencias indeseadas.

6.5.3. Atributos pertinentes de las entidades e interrelaciones

Las mayores dificultades para extender el tratamiento de la normalización del modelo relacional al modelo ER surgen de las distintas formas de expresar la identificación. Los identificadores externos de una entidad proveen dependencias funcionales entre atributos de la entidad. De forma similar, dada una interrelación entre varias entidades, los identificadores de las entidades proveen dependencias funcionales entre atributos de la interrelación. Por estas razones se necesita definir, para cada entidad o interrelación, el conjunto de atributos que son *pertinentes* para el proceso de normalización; este conjunto contiene los atributos originales de la entidad o interrelación, pero por lo general es más grande. Se empieza por considerar la identificación externa.

Atributos pertinentes de una entidad. Considérese una entidad E; para cada identificación externa a partir de una entidad E_1 , se incorpora en el conjunto de atributos pertinentes de E un identificador interno de E_1 . Este proceso es iterativo si a su vez E_1 posee identificadores externos. Por ejemplo, considérese la entidad detalle-pedido de la figura 6.14a, con los atributos NUM_DE_LINEA, CANTIDAD, PRECIO_POR_UNIDAD y los identificadores externos provistos por la entidad PEDIDO. De acuerdo con la definición anterior, los atributos pertinentes de DETALLE_PEDIDO son NUM_DE_PEDIDO, NUM_DE_PEDIDO, CANTIDAD y PRECIO_POR_UNIDAD.

Nótese que, como resultado de incluir NUM_DE_PEDIDO entre los atributos pertinentes, se puede ahora considerar el par (NUM_DE_PEDIDO, NUM_DE_LINEA) como un identificador de la entidad DETALLE_PEDIDO.

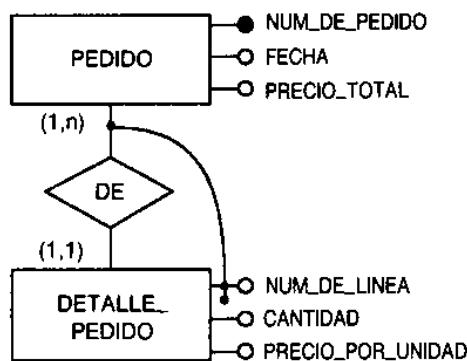
En el resto de este capítulo, simplemente se denotará como identificador de una entidad a cualquier subconjunto de los atributos pertinentes que identifique de manera única los casos de la entidad, sin tener en cuenta la diferencia entre identificación externa e interna.

Atributos pertinentes de una interrelación. Sea R una interrelación binaria entre las entidades E_1 y E_2 ; m_1 y m_2 denotan las cardinalidades máximas de E_1 y E_2 , respectivamente, en R. Se consideran tres casos:

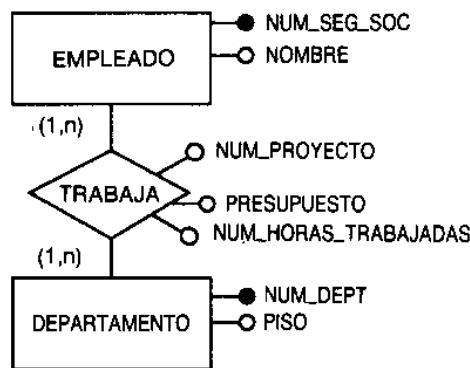
1. *Interrelación de uno a uno* ($m_1 = 1$ y $m_2 = 1$). Se incorpora en los atributos pertinentes de R un identificador elegido arbitrariamente, sea de E_1 o de E_2 .
2. *Interrelación de uno a muchos* ($m_1 = 1$ y $m_2 = n$). Se incorpora en los atributos pertinentes de R un identificador de E_1 .
3. *Interrelación de muchos a muchos* ($m_1 = n$ y $m_2 = n$). Se incorpora en los atributos pertinentes de R un identificador de E_1 y uno de E_2 .

Estas reglas se extienden con facilidad a las interrelaciones n-arias.

Considérese el ejemplo de la figura 6.14b, que describe las entidades EMPLEADO y DEPARTAMENTO y la interrelación de muchos a muchos TRABAJA, entre ellas. En este caso, los identificadores NUM_SEG_SOC para EMPLEADO y NUM-



(a) Atributos pertinentes de una entidad

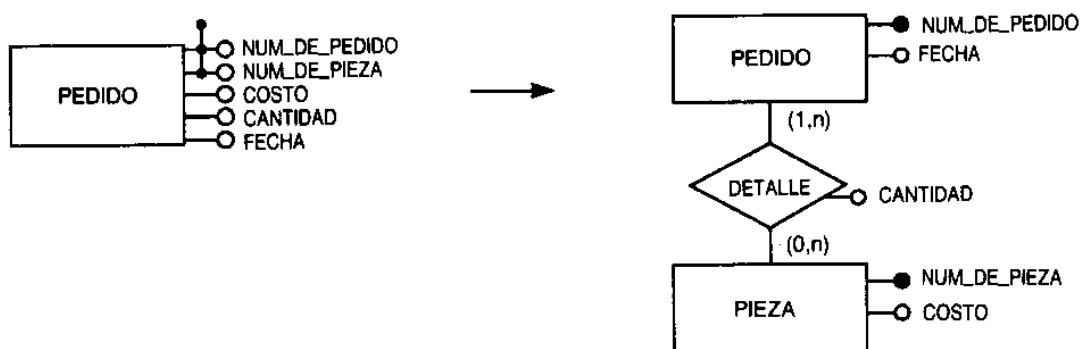


(b) Atributos pendientes de una interrelación

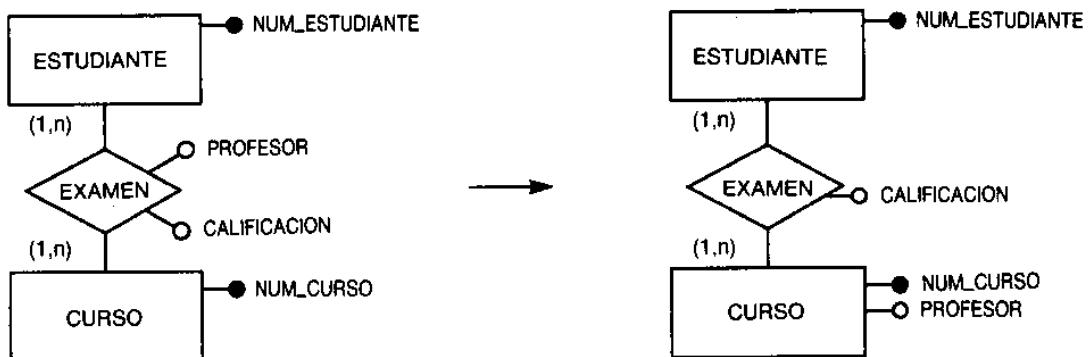
Figura 6.14. Atributos pertinentes.

_DEPT para **DEPARTAMENTO**, se incorporan en los atributos pertinentes de **TRABAJA**, que son: **NUM_SEG_SOC**, **NUM_DEPT**, **NUM_PROYECTO**, **PRESUPUESTO** y **NUM_HORAS_TRABAJADAS**.

Ahora se tienen las herramientas apropiadas para aplicar la normalización a los atributos ampliados de las entidades e interrelaciones. En la teoría relacional de la normalización, se definen varias formas normales progresivas (primera, segunda, tercera, Boyce-Codd, cuarta y otras). La primera forma normal la logra automáticamente cualquier relación *plana*; es decir, una sin entradas anidadas y en la cual todos los valores sean atómicos, lo que implica que los valores repetitivos o las listas no están permitidas. Cada normalización posterior impone la eliminación de ciertos tipos de anomalías. Estudiaremos la primera forma normal en el capítulo 12, en el contexto de la correspondencia del modelo ER al modelo relacional; en particular, se tratarán los atributos polivalentes, con cardinalidad máxima > 1 . En este capítulo se limita la atención a los atributos monovalentes. Cada entidad o interrelación, ampliada adecuadamente con los atributos pertinentes y considerada como aislada del resto del esquema, es equivalente estructuralmente a una relación plana y puede normalizarse.



(a) Primer ejemplo



(b) Segundo ejemplo

Figura 6.15. Cómo lograr la segunda forma normal.

6.5.4. Segunda forma normal

Antes de introducir la segunda forma normal, es necesario definir los atributos primos: un atributo no es **primo** (es no primo) si no pertenece a ningún identificador; es **primo** en caso contrario. Una entidad o interrelación está en **segunda forma normal** si no existe ninguna DF cuyo determinante esté propiamente contenido en un identificador y cuyo atributo del lado derecho sea no primo. Como ejemplo de entidad que viola la segunda forma normal considérese la entidad **PEDIDO**, de la figura 6.15a, en la que **NUM_DE_PIEZA** reemplaza a **NOMBRE_DE_PIEZA** de la figura 6.13. Aquí, el par (**NUM_DE_PEDIDO**, **NUM_DE_PIEZA**) es el único identificador. **COSTO**, **CANTIDAD** y **FECHA** son atributos no primos. Se definen las siguientes DF:

- $d_1: \text{NUM_DE_PEDIDO}, \text{NUM_DE_PIEZA} \rightarrow \text{CANTIDAD}$
- $d_2: \text{NUM_DE_PIEZA} \rightarrow \text{COSTO}$
- $d_3: \text{NUM_DE_PEDIDO} \rightarrow \text{FECHA}$

La entidad PEDIDO no está en segunda forma normal porque las dependencias d_2 y d_3 tienen un determinante que está contenido propiamente en el identificador, y COSTO y FECHA son no primos.

La violación de la segunda forma normal indica un mal diseño conceptual; en realidad, se ha representado en la entidad PEDIDO tres conceptos diferentes: pedidos, piezas y la interrelación entre ellos. Se logra la segunda forma normal al introducir exactamente un concepto del esquema por cada objeto del dominio de aplicación. El esquema resultante se muestra a la derecha de la figura 6.15a. La introducción de una entidad PIEZA origina la introducción de una interrelación muchos a muchos, DETALLE, entre PEDIDO y PIEZA.

Un ejemplo de segunda forma normal para interrelaciones se muestra en la figura 6.15b. En este caso, se tiene las dos entidades ESTUDIANTE y CURSO, y una interrelación EXAMEN entre ellas; los atributos pertinentes de EXAMEN son NUM_ESTUDIANTE, NUM_CURSO, PROFESOR y CALIFICACION. El par (NUM_ESTUDIANTE, NUM_CURSO) es el único identificador. Por tanto,

$$\text{NUM_ESTUDIANTE, NUM_CURSO} \rightarrow \text{PROFESOR, CALIFICACION}$$

Sin embargo, si cada curso tiene sólo un profesor, la siguiente DF se cumple: $\text{NUM_CURSO} \rightarrow \text{PROFESOR}$. Como PROFESOR es un atributo no primo, se viola la segunda forma normal. De nuevo, la violación de la normalización indica un mal diseño conceptual, que se puede mejorar trasladando el atributo PROFESOR a la entidad CURSO, como muestra la figura 6.15b.

6.5.5. Tercera forma normal

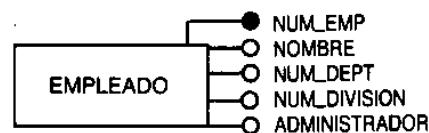
Antes de presentar la tercera forma normal, hay que definir las dependencias transitivas. Una DF, $A \rightarrow C$, es **transitiva** si existen dos dependencias $A \rightarrow B$, $B \rightarrow C$, tales que A, B y C sean grupos diferentes de atributos. Entonces, la dependencia $A \rightarrow C$ se puede inferir como una combinación de $A \rightarrow B$, $B \rightarrow C$; como tal, esta dependencia es redundante y una causa de anomalías. Considérese el ejemplo de la figura 6.16, que describe la entidad EMPLEADO con los atributos NUM_EMP, NOMBRE, NUM_DEPT, NUM_DIVISION, ADMINISTRADOR. NUM_EMP es el único identificador; por lo que se deduce la DF

$$\text{NUM_EMP} \rightarrow \text{NOMBRE, NUM_DEPT, NUM_DIVISION, ADMINISTRADOR}$$

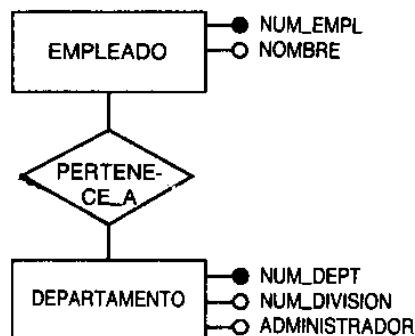
Se observa que cada departamento pertenece exactamente a una división y que cada división tiene un administrador; de aquí se deduce que existen las siguientes DF:

$$\begin{aligned} d_1: \text{NUM_DEPT} &\rightarrow \text{NUM_DIVISION} \\ d_2: \text{NUM_DIVISION} &\rightarrow \text{ADMINISTRADOR} \end{aligned}$$

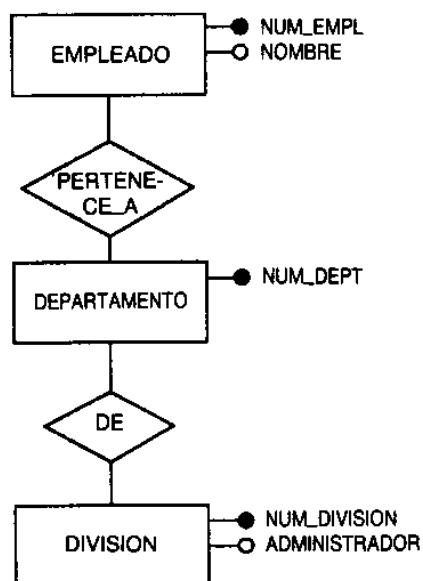
Se deducen, entonces, varias dependencias transitivas:



(a) Primer esquema

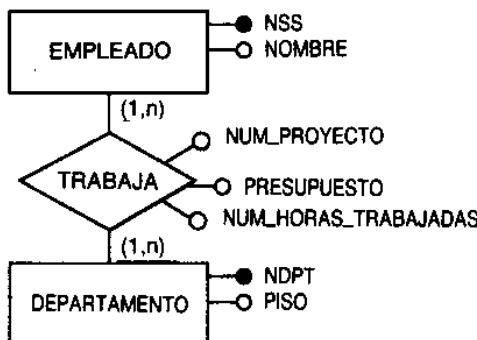


(b) Segundo esquema

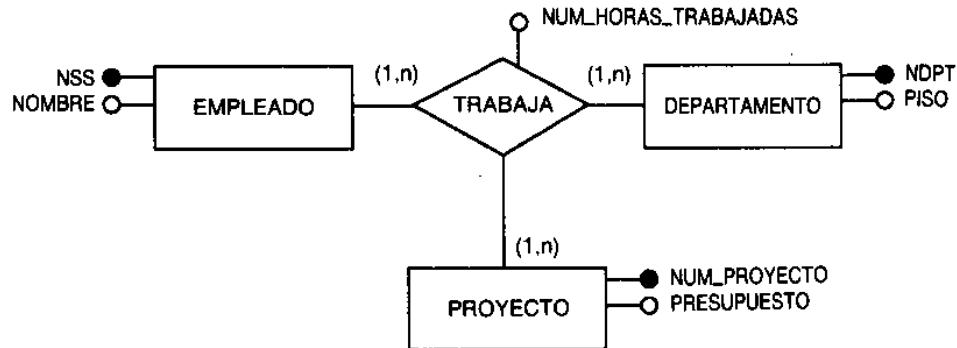


(c) Tercer esquema

Figura 6.16. Logro de la tercera forma normal: primer caso.



(a) Primer esquema



(b) Segundo esquema

Figura 6.17. Logro de la tercera forma normal: segundo caso.

$\text{NUM_EMP} \rightarrow \text{NUM_DIVISION}, \text{ADMINISTRADOR}$
 $\text{NUM_DEPT} \rightarrow \text{ADMINISTRADOR}$

Una entidad o interrelación está en **tercera forma normal** si se encuentra en segunda forma normal y no tiene ninguna DF transitiva. Claramente, la entidad EMPLEADO del ejemplo viola la tercera forma normal. Una vez más, la violación de la normalización indica un mal diseño, ya que la entidad EMPLEADO incorpora los conceptos de DEPARTAMENTO y DIVISION. La normalización progresiva de EMPLEADO se lleva a cabo en las figuras 6.16b y 6.16c. Se introducen dos nuevas entidades, cada una de ellas correspondiente a un concepto diferente de la realidad; de esta manera, se eliminan las dependencias transitivas.

Como otro ejemplo de una interrelación que viola la tercera forma normal, considérese la interrelación TRABAJA de la figura 6.17. Puesto que el par (NSS, NDPT) es el identificador de TRABAJA, se deducen las siguientes dependencias:

$d_1: \text{NSS, NDPT} \rightarrow \text{PRESUPUESTO}$
 $d_2: \text{NSS, NDPT} \rightarrow \text{NUM_PROY}$

$d_3: NSS, NDPT \rightarrow NUM_HORAS_TRABAJADAS$

Sin embargo, cada proyecto (con su **NUM_PROY** asociado) tiene un presupuesto único; de aquí deducimos la dependencia

$d_4: NUM_PROY \rightarrow PRESUPUESTO$

Por tanto, la dependencia d_1 es transitiva y la interrelación no está en tercera forma normal. Se puede lograr la tercera forma normal introduciendo la nueva entidad **PROYECTO** y transformando la interrelación **TRABAJA** en una interrelación ternaria (véase la figura 6.17b). Nótese que, una vez más, la normalización se logra al introducir un concepto aparte por cada objeto del dominio de aplicación.

6.5.6. Forma normal de Boyce-Codd

Otra forma normal muy conocida en el modelo relacional es la forma normal de Boyce-Codd. Una entidad o interrelación está en la **forma normal de Boyce-Codd (BCNF)**, si cada determinante de sus DF es un identificador. Un ejemplo de relación en BCNF es la entidad **ESTUDIANTE**, con dos identificadores, número de seguro social (NSS) y número de estudiante (NESTU), y dos atributos no primos, **NOMBRE** y **DIRECCION**. En este caso, se tienen las dependencias

$d_1: NSS \rightarrow NESTU, NOMBRE, DIRECCION$
 $d_2: NESTU \rightarrow NSS, NOMBRE, DIRECCION$

Estas DF no violan la BCNF.

Considérese ahora el esquema de la figura 6.18, que representa estudiantes, cursos y profesores; supóngase que cada profesor imparte exactamente un curso, que el mismo curso puede ser impartido por varios profesores, y que cada estudiante asiste a un determinado curso con un único profesor. Las restricciones antes señaladas pueden expresarse en términos de las siguientes DF:

$d_1: ESTUDIANTE, NCURSO \rightarrow PROFESOR$
 $d_2: PROFESOR \rightarrow NCURSO$

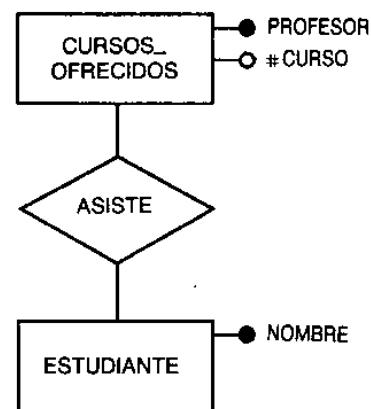
La entidad **ESTRUCTURA_DE_CURSOS** tiene dos identificadores alternativos: el par (**ESTUDIANTE, NCURSO**) y el par (**ESTUDIANTE, PROFESOR**). Entonces, la DF d_2 viola la BCNF, ya que **PROFESOR** es un determinante pero no un identificador.

Ya que la violación de la BCNF se sigue del hecho de que **PROFESOR** no es un determinante, se puede tratar de separar el concepto de **PROFESOR** y el **NCURSO** asociado, que juntos se denominan una oferta de curso, del concepto de **ESTUDIANTE**; esta transformación se muestra en la figura 6.18b. Sin embargo, la in-

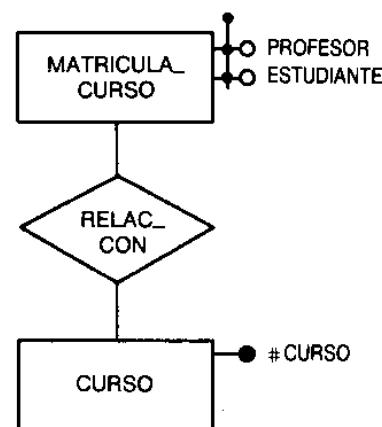


Dependencias funcionales:
 $\text{ESTUDIANTE}, \text{NCURSO} \rightarrow \text{PROFESOR}$
 $\text{PROFESOR} \rightarrow \text{NCURSO}$

(a) Esquema inicial



(b) Primera solución



(c) Segunda solución

Figura 6.18. Intentos de lograr la forma normal de Boyce-Codd.

La relación ASISTE es equivalente a la entidad original ESTRUCTURA_DE_CURSOS y, a su vez, viola la BCNF.

Como una segunda solución, se puede separar los conceptos de PROFESOR y

ESTUDIANTE del concepto de CURSO, introduciendo una entidad MATRICULA_CURSO con los atributos PROFESOR y ESTUDIANTE (Fig. 6.18c). En este caso, surge un problema nuevo, pues la dependencia funcional PROFESOR → NCURSO ya no está representada en el esquema. No es posible deducirla ni de las entidades e interrelaciones internas ni de los identificadores externos. De hecho, d_2 no es una dependencia ni de la entidad MATRICULA_CURSO, ni de la entidad CURSO, ni de la interrelación RELACIONADO_CON. Así pues, el esquema nuevo no puede considerarse equivalente al antiguo, porque se ha perdido una dependencia.

Todas las posibles soluciones reproducen la violación o implican la pérdida de una dependencia; por tanto, no hay forma de transformar el esquema en uno nuevo en BCNF que mantenga dependencias. Este es el equivalente, en el modelo ER, de un conocido resultado de la teoría de la normalización relacional, a saber, que algunos esquemas no se pueden llevar a la forma normal de Boyce-Codd manteniendo las dependencias funcionales. Como regla general, se debe tratar siempre de lograr la forma normal de Boyce-Codd, si ello es posible, pero esto es algo que *no* se puede lograr en todos los casos.

6.5.7. Cómo razonar respecto a la normalización

Las formas normales subsecuentes requieren la identificación y el empleo de nuevos tipos de dependencias (como las dependencias multivaluadas, de dominio, de reunión y de inclusión). El enfoque utilizado en los apartados anteriores puede ampliarse para tratar estos tipos de dependencias, pero no se tratarán en este libro; más bien, se evaluará la importancia de la normalización en el marco del diseño conceptual de bases de datos.

En el enfoque de este libro, la normalización es una *herramienta para validar la calidad* del esquema, más que *un método para diseñar* el esquema. Esto es discutible, porque varios enfoques proponen la normalización como el único método aceptable para diseñar bases de datos. Estos enfoques sugieren el uso del propio modelo relacional como modelo de diseño; suponen un conjunto de DF como entrada al proceso de diseño y producen una serie de relaciones en una forma normal dada como resultado.

Aunque creemos que la normalidad es una propiedad importante de un esquema, pensamos que las dependencias son medios inadecuados y poco claros para captar los requerimientos en el dominio de la aplicación. Preferimos los mecanismos de abstracción que ofrece el modelo ER, pues son fáciles de entender y se pueden representar mediante diagramas en forma natural.

Al mismo tiempo, se subraya que el modelo ER y los métodos de diseño descritos en el capítulo 3 *tienden a producir, con naturalidad, esquemas normalizados*. El objetivo de la normalización, hasta la forma normal de Boyce-Codd, es mantener las dependencias funcionales separadas, al asociar con cada grupo de DF homogéneas un elemento del modelo (entidad o interrelación) que tenga los determinantes de las DF como identificadores. Así, cada concepto del dominio de aplicación corresponde exactamente a un concepto del esquema.

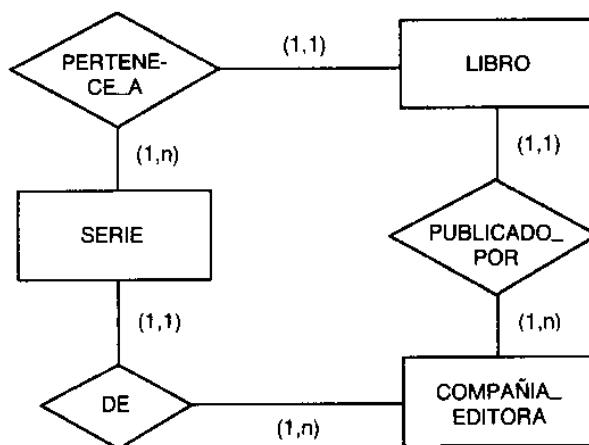


Figura 6.19. Redundancia en el ejemplo de la figura 5.8.

Esta *separación de conceptos* es el resultado natural del uso de abstracciones de clasificación, agregación y generalización, y de la aplicación de una metodología descendente para producir un esquema mediante refinamientos sucesivos. En todos los ejemplos que violan la normalización, se ha observado que el esquema correspondiente estuvo mal diseñado. Esto concuerda con nuestra visión de la normalización como herramienta de validación.

6.6. Ejemplo de reestructuración de un esquema

Considérese de nuevo el esquema obtenido al final del capítulo 5 (Fig. 5.8), como resultado de la actividad de integración, para verificar su calidad.

Considerando primero la minimalidad, se descubre un ciclo redundante de interrelaciones (Fig. 6.19): cada libro pertenece a una serie de una determinada compañía editora; por tanto, se puede deducir la interrelación **PUBLICADO_POR** entre **LIBRO** y **COMPAÑIA_EDITORA** como la combinación de las dos interrelaciones **PERTENECE_A** y **DE**. La decisión de mantener **PUBLICADO_POR** en el esquema o no depende de su importancia en la aplicación: aquí hay un conflicto entre expresividad y minimalidad. En ausencia de argumentos contundentes, se puede eliminar la fuente de redundancia o simplemente indicarla en el esquema conceptual, posponiendo la decisión hasta la fase de diseño lógico (véase el capítulo 11).

Luego se considera la expresividad. Se puede aplicar dos transformaciones. Primero, considérese el fragmento del esquema en la figura 6.20. Se nota que **LIBRO** y **ARTICULO** comparten las mismas interrelaciones con **AUTOR** y **CIENTIFICO**. La expresividad global mejora al introducir un concepto más general que **LIBRO** y **ARTICULO**, llamado **TRABAJO_AUTOR**, que hereda las interrelaciones

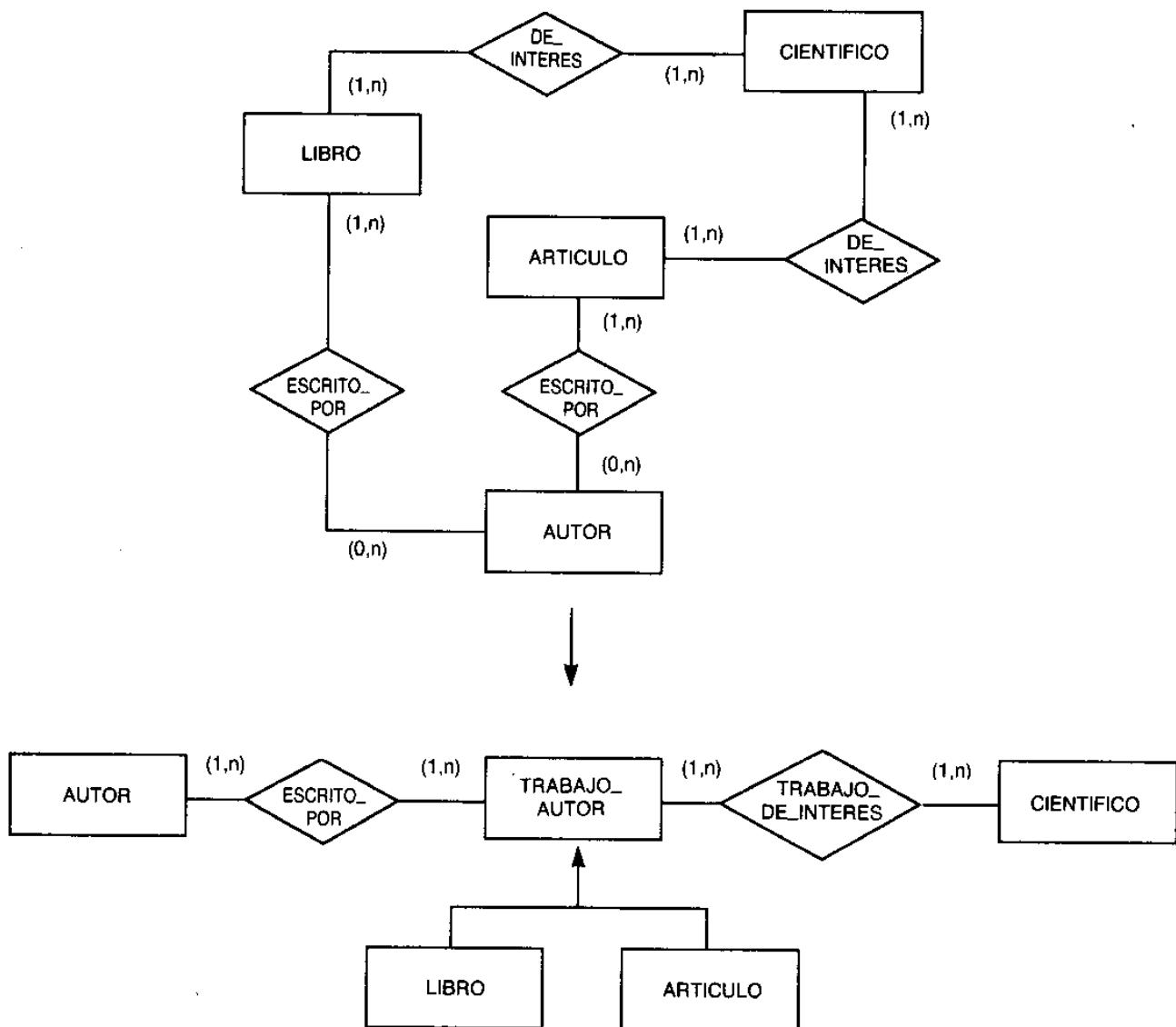


Figura 6.20. Mejoramiento de la expresividad y la autoexplicación en el ejemplo.

anteriores. Nótese que, de esta forma, LIBRO es una entidad subconjunto tanto de PUBLICACION como de TRABAJO_AUTOR (técnicamente, esta situación suele denominarse *herencia múltiple*). Esta transformación mejora también la autoexplicación: en el esquema anterior, la participación de la entidad AUTOR era opcional tanto en la interrelación con la entidad LIBRO como en la interrelación con la entidad ARTICULO; sin embargo, se necesitaba una restricción adicional que afirmara que todo autor tiene que estar conectado con un libro o bien con un artículo. Esta restricción no estaba expresada explícitamente en el esquema. Después de la reestructuración, se cambia a 1 el valor de cardinalidad mínima de la entidad AUTOR en la interrelación ESCRITO_POR.

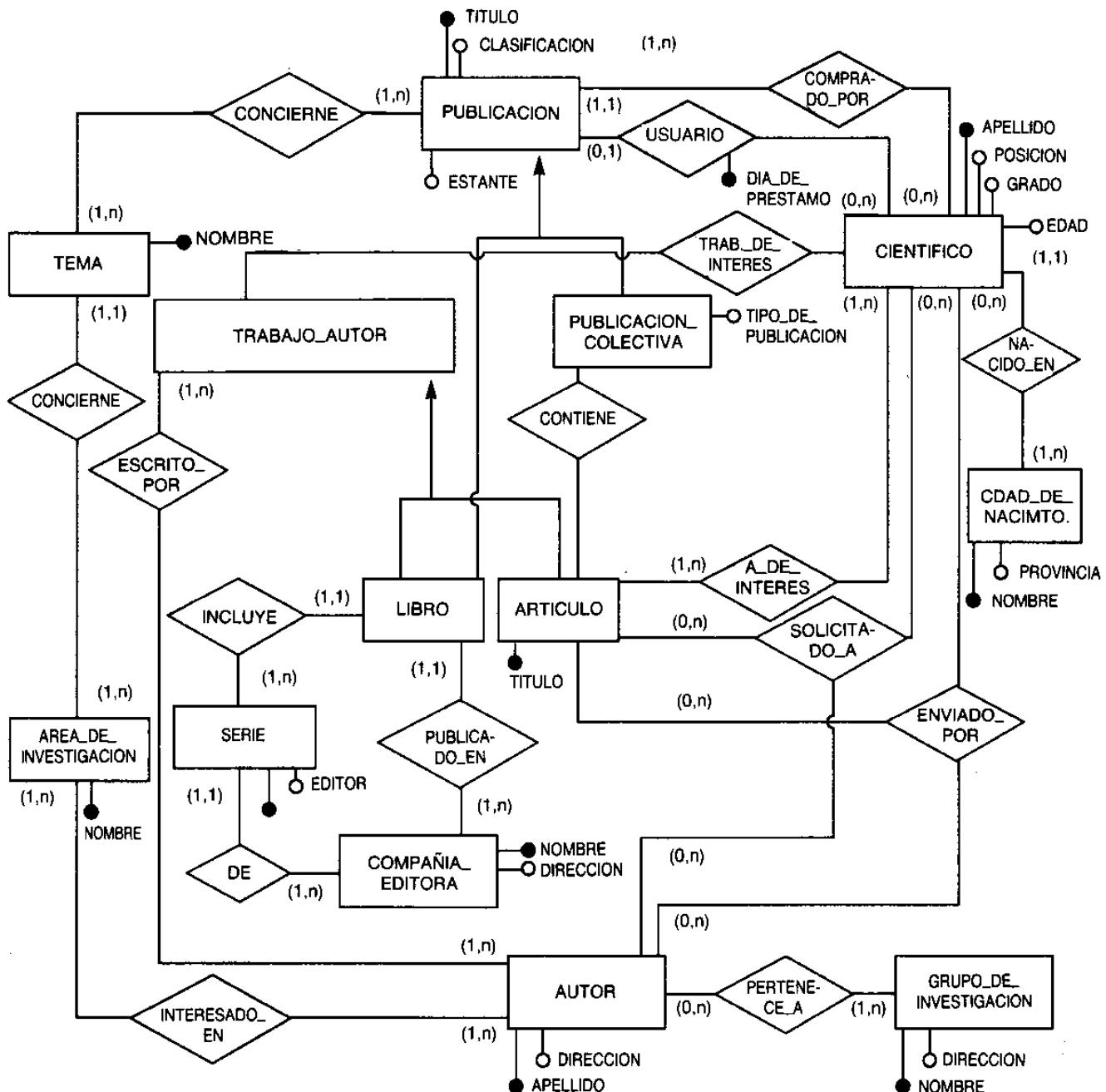


Figura 6.21. Esquema global después de la reestructuración.

Otra acción de reestructuración se puede aplicar al fragmento del esquema que incluye las entidades REVISTA y ANALES (Fig. 5.8). Al tener estas entidades las mismas propiedades, pueden fusionarse en una única entidad, llamada PUBLICACION_COLECTIVA, con el atributo adicional TIPO_DE_PUBLICACION, (véase el esquema final en la figura 6.21).

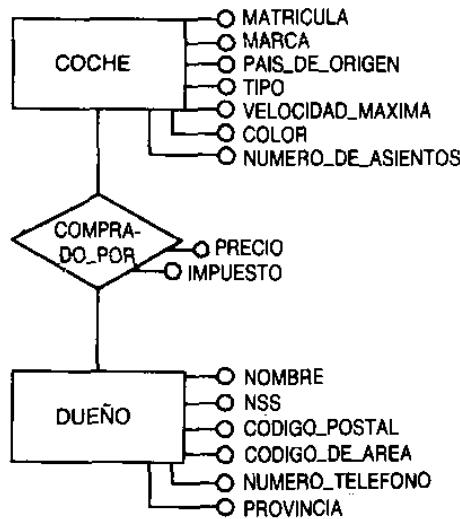


Figura 6.22. Base de datos propietarios de coches.

Finalmente, se considera la normalización. La única violación de la normalización se observa en la entidad CIENTIFICO: si se supone, como de costumbre, que CIUDAD_DE_NACIMIENTO determina PROVINCIA_DE_NACIMIENTO, la entidad no estará en tercera forma normal. La normalización se obtiene introduciendo la nueva entidad CIUDAD_DE_NACIMIENTO, con los atributos NOMBRE y PROVINCIA. La figura 6.21 muestra el esquema conceptual final después de la reestructuración.

6.7. Resumen

Las cualidades de un esquema conceptual son compleción, corrección, minimalidad, expresividad, legibilidad, autoexplicación, extensibilidad y normalidad. Estas cualidades deben guiar el proceso de diseño; deben comprobarse varias veces mientras se diseña una base de datos y deben considerarse cuidadosamente al final del diseño conceptual. En particular, se presentan varias técnicas para mejorar la minimalidad, expresividad, autoexplicación y normalidad; estas técnicas se basan en el uso de transformaciones que deben aplicarse a los esquemas.

Las transformaciones para el logro de la normalidad se aplican a las entidades para eliminar las anomalías de actualización. Este proceso se basa en el uso de dependencias funcionales. La normalización se desarrolló originalmente para el modelo de datos relacional, y se usó como técnica autónoma para construir esquemas relacionales. En este libro, la técnica se aplica al modelo ER, pero se usa para reestructurar esquemas más que para construirlos. Sin embargo, las

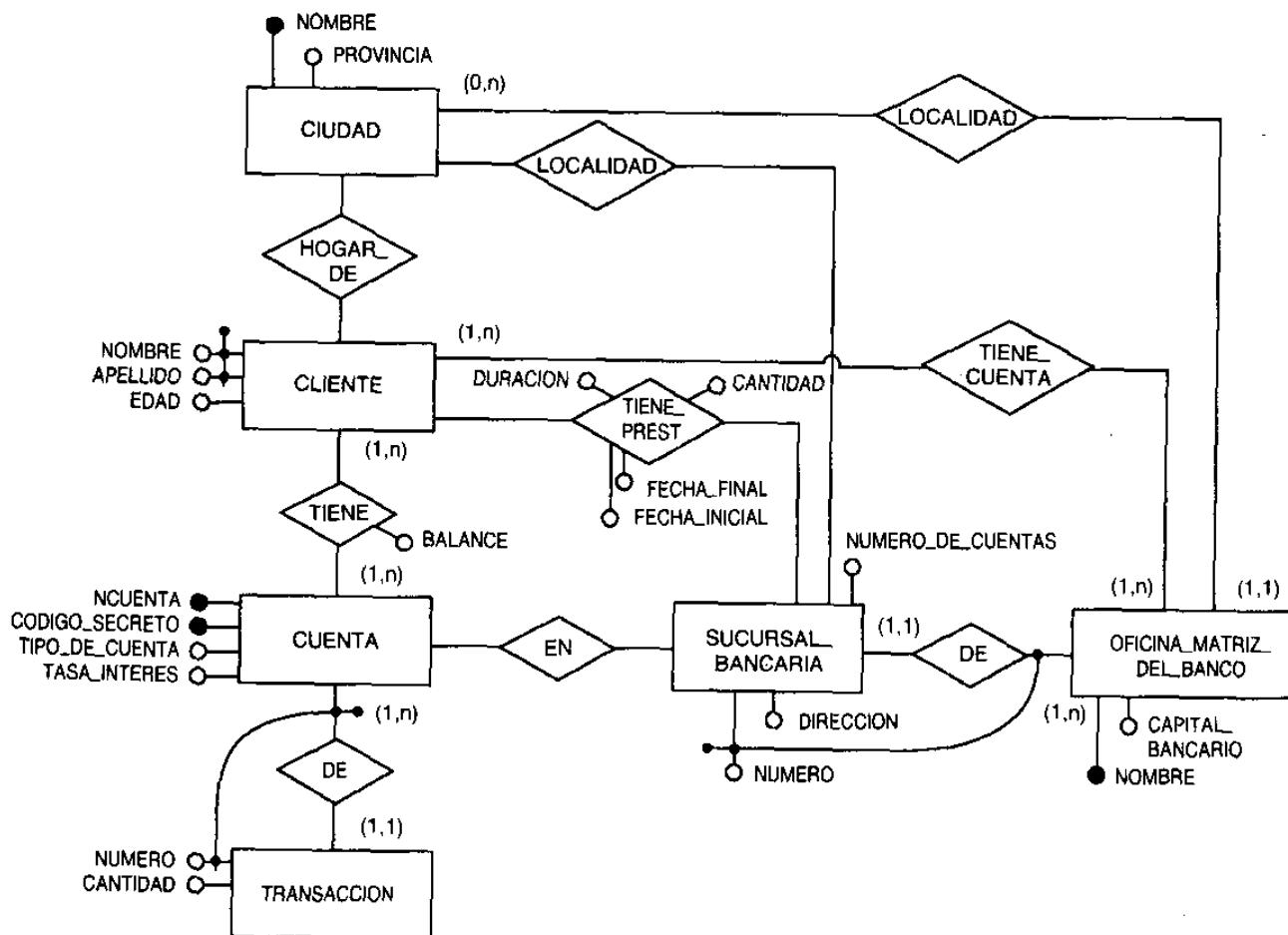


Figura 6.23. Base de datos de un banco.

operaciones de reestructuración que concentran la atención en la normalización y los conceptos de tercera forma normal y forma normal de Boyce-Codd son muy importantes y útiles para los diseñadores de bases de datos.

Ejercicios

- 6.1. Extienda las reglas para buscar atributos pertinentes de interrelaciones binarias (véase el apartado 6.6) a las interrelaciones n-arias.
- 6.2. Considere los esquemas producidos para los ejercicios 5.3 y 5.4 y revise sus cualidades. Escriba una evaluación de cada calidad.
- 6.3. Revise la expresividad del esquema del impuesto sobre la renta de la figura 4.16.
- 6.4. Revise la normalidad del esquema del impuesto sobre la renta. (Nótese que

- la presencia de datos derivados produce un gran número de dependencias funcionales.)
- 6.5. Revise la normalidad del esquema de la figura 6.22. Haga todas las suposiciones razonables que necesite y enúncielas.
 - 6.6. Revise la redundancia y normalidad del esquema de la figura 6.23. Haga todas las suposiciones razonables que necesite y enúncielas.

Bibliografía

- P. Bernstein, «Synthesizing Third Normal Form Relations from Functional Dependencies», *ACM Transactions on Database Systems*, 1, núm. 4, 1976.
E.F. Codd. «Further Normalization of the Database Relational Model». En R. Rustin, ed., *Database Systems*, Prentice-Hall, 1972.
R. Fagin. «The Decomposition versus the Synthetic Approach to Relational Database Design». En *Proc. Third International Conference on Very Large Databases*, Tokio, 1977.

Estos artículos se mencionan aquí por razones históricas y por su importancia al ofrecer un tratamiento formal de las formas normales. El artículo pionero de Codd introduce las formas normales. El artículo de Bernstein describe un procedimiento de síntesis. A partir de un conjunto de atributos y de dependencias funcionales, ofrece un algoritmo para sintetizar relaciones en tercera forma normal. El artículo de Fagin parte de las mismas entradas y descompone una relación universal inicial, formada por todos los atributos. Las reglas de descomposición están guiadas por las dependencias funcionales.

Los dos enfoques, el de Fagin y el de Bernstein, pueden clasificarse como estrategias de diseño ascendentes, porque toman como punto de partida el conjunto completo de atributos y dependencias funcionales. Un largo debate se ha centrado en los pros y contras del modelado conceptual frente al uso de la normalización basado en el modelo relacional. La experiencia en el diseño de grandes bases de datos ha mostrado que el modelado conceptual es, en realidad, mucho más efectivo; muchas metodologías modernas de diseño, incluyendo la de este libro, utilizan la normalización como técnica de verificación para los esquemas conceptuales.

- C. Beeri, P.A. Bernstein y N. Goodman, «A Sophisticate's Introduction to Database Normalization Theory». En *Proc. Fourth International Conference on Very Large Databases*, Berlin, 1978.

Este artículo es un estudio completo de la teoría de la normalización.

- A.V. Aho, C. Beeri y J.D. Ullman, «The Theory of Joins in Relational Databases». *ACM Transaction on Database Systems*, 4, núm. 3, 1979, 297-314.

- C. Beeri, «On the Membership Problem for Functional and Multivalued Dependencies in Relational Databases», *ACM Transaction on Database Systems*, 5, núm. 3, sept. 1980, 241-59.

- C. Beeri y P.A. Bernstein, «Computational Problems Related to the Design of Normal Form Relational Schemas», *ACM Transaction on Database Systems*, 4, núm. 1, mar. 1979, 30-59.

- C. Zaniolo y M.A. Melkanoff, «On the Design of Relational Database Schemata», *ACM Transaction on Database Systems*, 6, núm. 1, mar. 1981, 1-47.

Estos artículos describen la complejidad computacional de la construcción de relaciones normalizadas. Beeri trata las dependencias multivaluadas y la cuarta forma normal, que no se consideran en este libro. Aho, Beeri y Ullman estudian las dependencias de reunión, un tipo de dependencia general, no considerada en este libro, y la propiedad de reunión sin pérdidas de las descomposiciones.

- W. Kent, «A Simple Guide to Five Normal Forms in Relational Database Theory», *Communications of the ACM*, 26, núm. 2, feb. 1983, 120-25.

Este artículo ofrece una introducción de orientación práctica a las formas normales.

- R. Brown y D.S. Parker, «LAURA: a Formal Data Model and Her Logical Design Methodology». En *Proc. Ninth International Conference on Very Large Databases*, Florencia, 1983.

- D. Embley y T. Ling, «Synergistic Database Design with an Extended Entity-Relationship Model». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1988.

Estos artículos presentan un enfoque de normalización similar al seguido en el apartado 6.6. El esquema ER, en vez de transformarse en un esquema relacional y luego normalizarse, primero se transforma en un esquema normalizado, lo que garantiza la generación de un esquema relacional normalizado.

- S. Ceri y G. Gottlob, «Normalization of Relations and PROLOG», *Communications of the ACM*, 29, núm. 6, jun. 1986, 524-44.

Este artículo ofrece una definición autónoma de las formas normales y una especificación en PROLOG de los algoritmos de normalización; también se incluye el código PROLOG de los algoritmos de normalización.

Documentación y mantenimiento de esquemas

Los capítulos anteriores han versado sobre el problema de diseñar esquemas, pero en este capítulo se centrará la atención en el problema de mantenerlos y guardarlos en un diccionario. Como un esquema no puede ser mantenido sin una documentación clara de su contenido de información, se abordará el tema de cómo debe hacerse la documentación de los esquemas.

Una buena documentación del diseño conceptual se propone conseguir que el proceso de generación del esquema conceptual a partir de los requerimientos quede tan claro como sea posible. En el capítulo 3 se analizaron las primitivas y estrategias que se pueden usar en el proceso de diseño conceptual, y se observó que aunque el enfoque descendente tiene varias ventajas sobre otros enfoques, su aplicación durante el proceso de diseño es algunas veces muy difícil. Sin embargo, un planteamiento descendente puro se puede usar a posteriori; la documentación tiene como resultado un conjunto de refinamientos que reproducen el proceso de generación entero paso a paso. Una organización descendente de la documentación ayuda a las actividades de mantenimiento subsiguientes: cuando los requerimientos cambian, la documentación descendente se puede usar de nuevo (reusar), y el diseño puede continuar simplemente modificando los pasos de refinamiento previos y adaptándolos a los nuevos requerimientos. Esto significa que los cambios en los requerimientos de una aplicación deben incorporarse primero en el esquema conceptual global y luego ser propagados a esquemas de niveles más bajos.

Conforme más y más aplicaciones se automatizan, las actividades de mantenimiento ganan importancia en el ciclo de vida de los sistemas de información; al mismo tiempo, se torna importante para una organización representar en un diccionario todos los esquemas generados en el pasado, para tener una representación estructurada del contenido completo de información de la organización.

Este capítulo se organiza de la siguiente manera. En el apartado 7.1 se habla

de la documentación y se proponen varios criterios para determinar niveles (planos) de refinamiento y verificar su calidad. En el apartado 7.2 se trata el mantenimiento y se describe una metodología que usa la documentación descendente como herramienta para mantener y actualizar esquemas conceptuales. Finalmente, el apartado 7.3 aborda el problema de estructurar y construir un diccionario de datos, visto como el depósito integrado de los metadatos de una organización.

7.1. Una metodología para documentar esquemas conceptuales

Incluso si el esquema conceptual ha sido producido siguiendo un enfoque que no es descendente puro, resulta útil producir su documentación usando un enfoque descendente puro. La documentación conceptual debe incluir lo siguiente:

1. El **esquema conceptual global** final de la aplicación.
2. Un conjunto ordenado de esquemas (denominados **planos de refinamiento**) que representan los requerimientos a varios niveles de refinamiento, de manera tal que para cada par de esquemas adyacentes S_1 y S_2 , S_2 es más detallado que S_1 en la representación de los requerimientos.
3. Para cada par de esquemas adyacentes S_1 y S_2 , un conjunto de transformaciones que hagan corresponder los conceptos de S_1 con los conceptos de S_2 . Se puede decir que las transformaciones describen cómo se obtiene S_2 partiendo de S_1 .

Los planos de refinamiento y las transformaciones son descripciones complementarias, ya que una se puede obtener de la otra. Al mismo tiempo, describen el proceso de generación desde dos puntos de vista diferentes: ésta es la razón para incluirlos ambos en la documentación.

Una metodología para la actividad de documentación se muestra en la figura 7.1. Dado el esquema global, se debe escoger primero un *esquema armazón* que represente el esquema global a un nivel abstracto; luego, se construye un conjunto de planos de refinamiento, es decir, nuevos esquemas a niveles más bajos de detalle, hasta producir el esquema final. Obsérvese que esta metodología es aplicable estrictamente sólo a esquemas que puedan ser producidos por refinamientos descendentes puros. En el capítulo 3 se mostró que algunos esquemas conceptuales no se pueden producir en forma puramente descendente; pero estos esquemas no son muy comunes en la práctica.

Se considerará ahora cómo escoger los planos de refinamiento y realizar una verificación final de calidad de la documentación. El proceso de construir un esquema abstracto S_0 a partir de un esquema más detallado S puede realizarse en dos pasos: 1) determinar una partición de S (cada componente de la partición es un subesquema de S), y 2) hacer corresponder cada subesquema de S

Entrada: S_0 , esquema conceptual global

Salida: Un conjunto ordenado de planos de refinamiento S_0, S_1, \dots, S_n , tal que cada plano se obtiene del anterior por medio de primitivas descendentes puras y $S_n = S_0$; para cada par consecutivo de planos S_i y S_{i+1} , un conjunto de primitivas de refinamiento que permite la generación de S_{i+1} a partir de S_i .

Encontrar una modularización de alto nivel del esquema S_0 , produciendo el esquema armazón S_0 .

Repetir.

Encontrar un esquema nuevo S_2 y un conjunto de primitivas descendentes ($A \rightarrow B$) tales que
 A, sea un concepto de S_0 ,
 B, sea un subesquema de S_2 , y
 S_2 se obtenga aplicando todas las primitivas a S_1 :

$S_1 \leftarrow S_2$

hasta que $S_2 = S_0$

Verificar las cualidades de los esquemas y refinamientos, y efectuar reestructuraciones si son necesarias

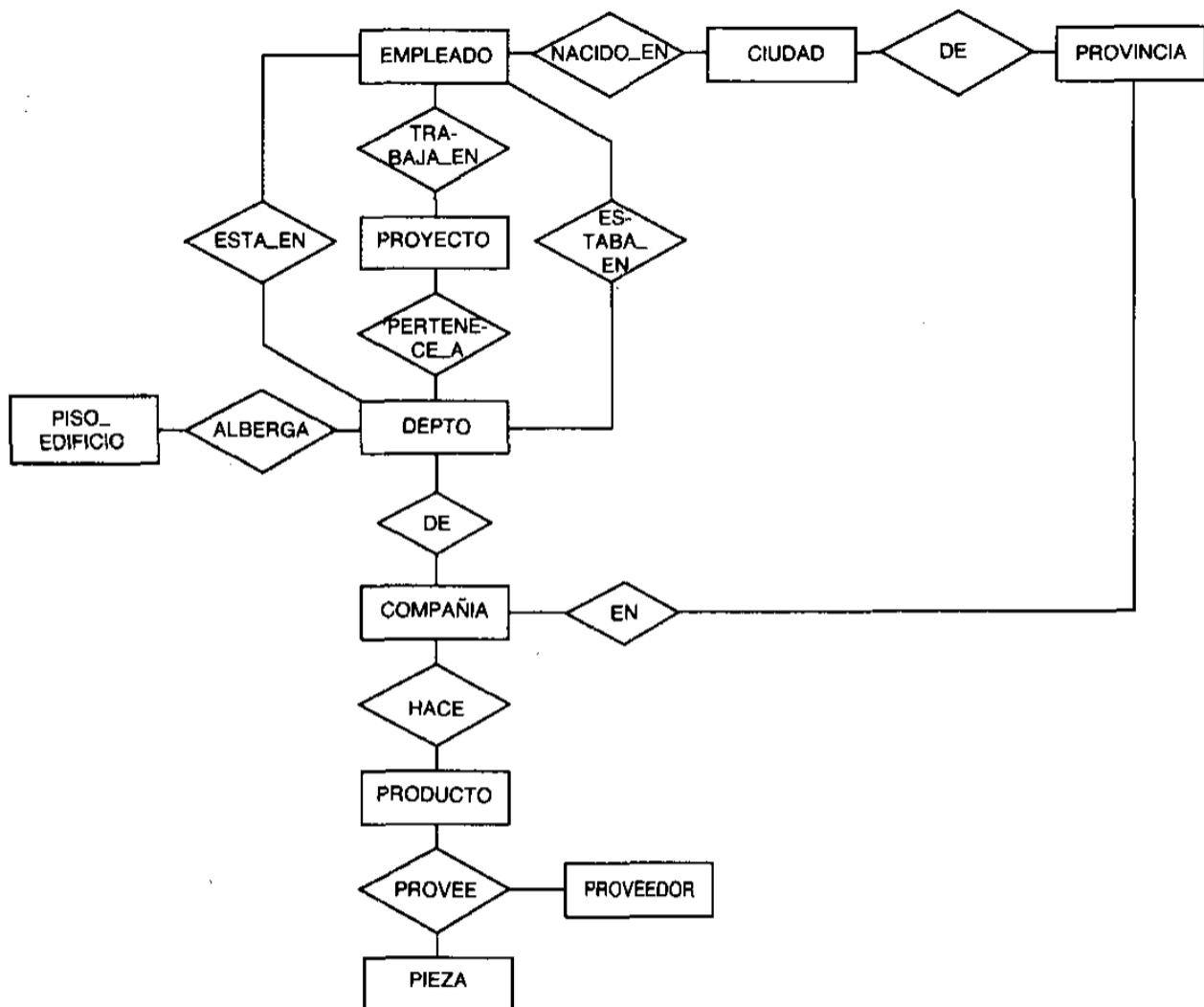
Figura 7.1. Una metodología para la documentación conceptual.

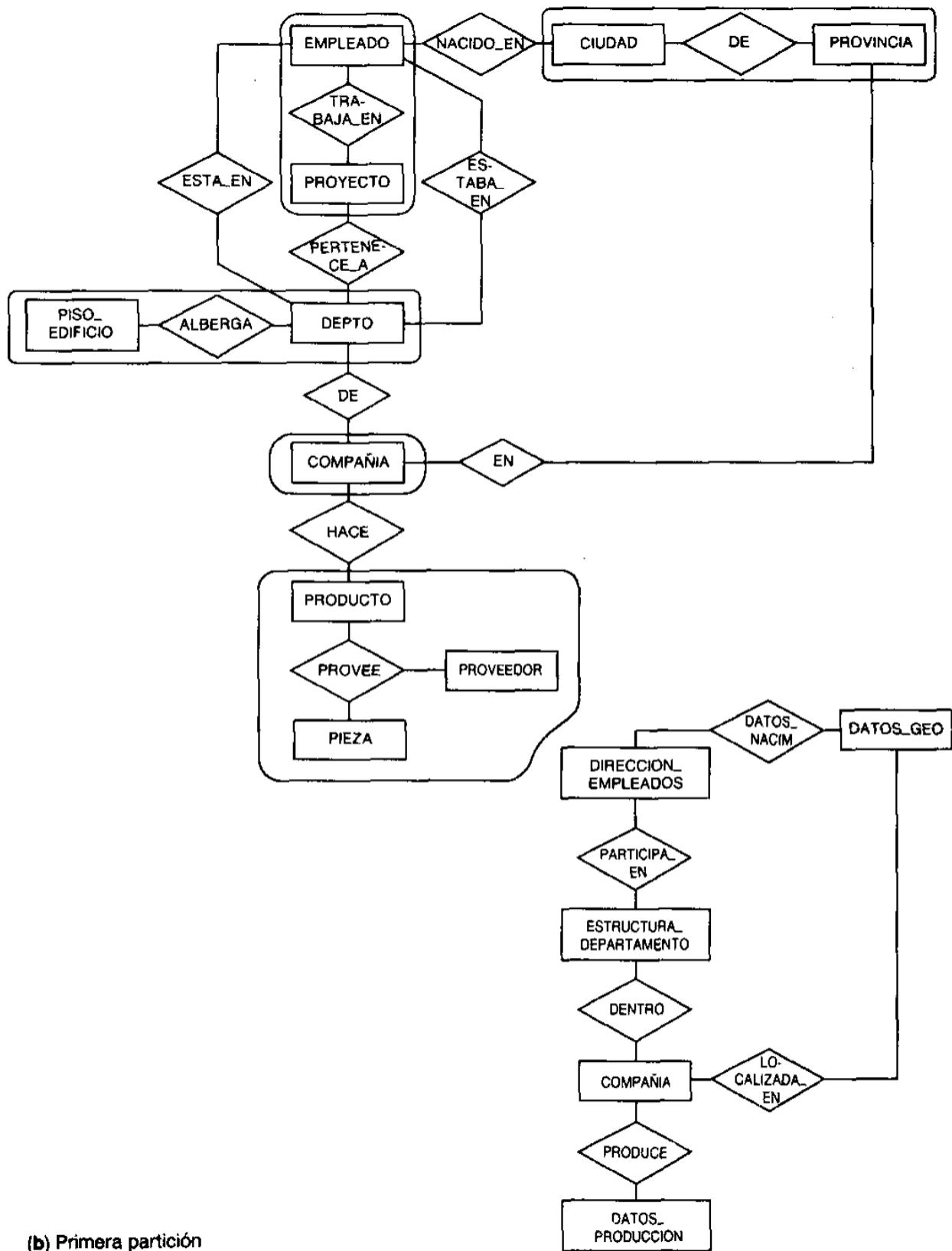
con un concepto de S_0 . Para determinar la partición, se debe seguir *criterios de modularización* similares a los usados en diseño de software; en concreto, se requieren las dos propiedades que siguen:

1. **Alta cohesión** entre los conceptos del mismo subesquema: los conceptos deben relacionarse con objetos del mundo real similares en el mismo subesquema.
2. **Bajo acoplamiento** entre los conceptos de subesquemas diferentes: las conexiones lógicas entre los componentes deben minimizarse.

En cierto sentido, la cohesión es un criterio semántico, mientras que el acoplamiento es más sintáctico. La cohesión debe detectarse teniendo en cuenta la aplicación específica. Por ejemplo, los conceptos que se refieren a productos de hardware y software pudieran estar estrechamente relacionados para una aplicación de mercado; pero pueden considerarse no relacionados para una aplicación de desarrollo de productos.

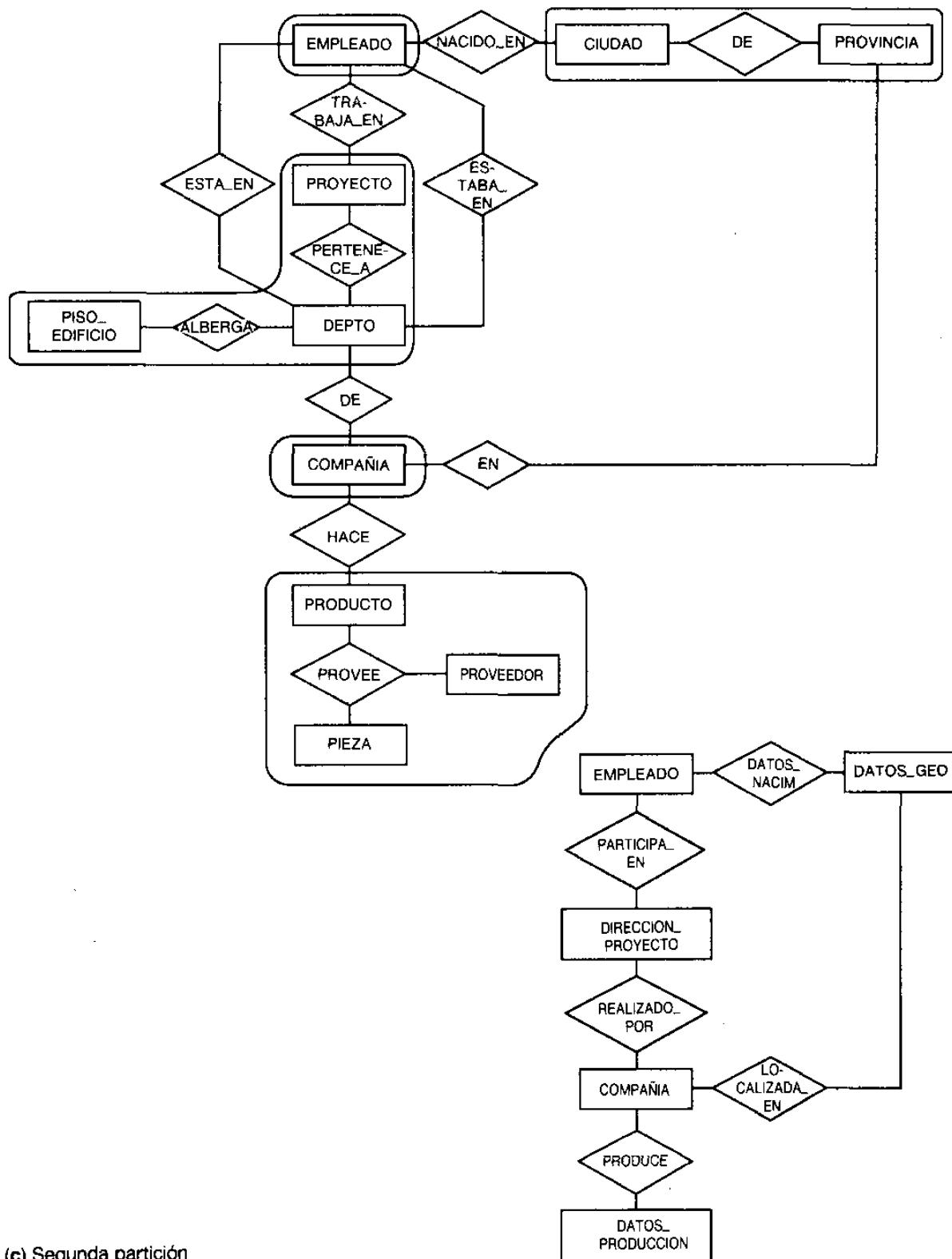
Como ejemplo de opciones alternativas para particiones, considérese el esquema de la figura 7.2a, que describe datos sobre compañías. Las figuras 7.2b, 7.2c, 7.2d muestran tres particiones diferentes. Tres subesquemas son comunes a todas las particiones, a saber, COMPAÑIA, DATOS_GEO, DATOS_PRODUCCION. Las tres particiones difieren en la manera en que agrupan las entidades COMPAÑIA, EMPLEADO, PROYECTO, DEPTO y PISO. La tercera partición es ligeramente mejor que la primera y la segunda porque tiene una cohesión más alta de conceptos y usa menos particiones (cuatro en lugar de cinco); además, cada partición es aproximadamente del mismo tamaño, lo que contribuye a un me-

(a) Esquema inicial (esquema conceptual global S_g).**Figura 7.2.** Ejemplos de elecciones de particiones.



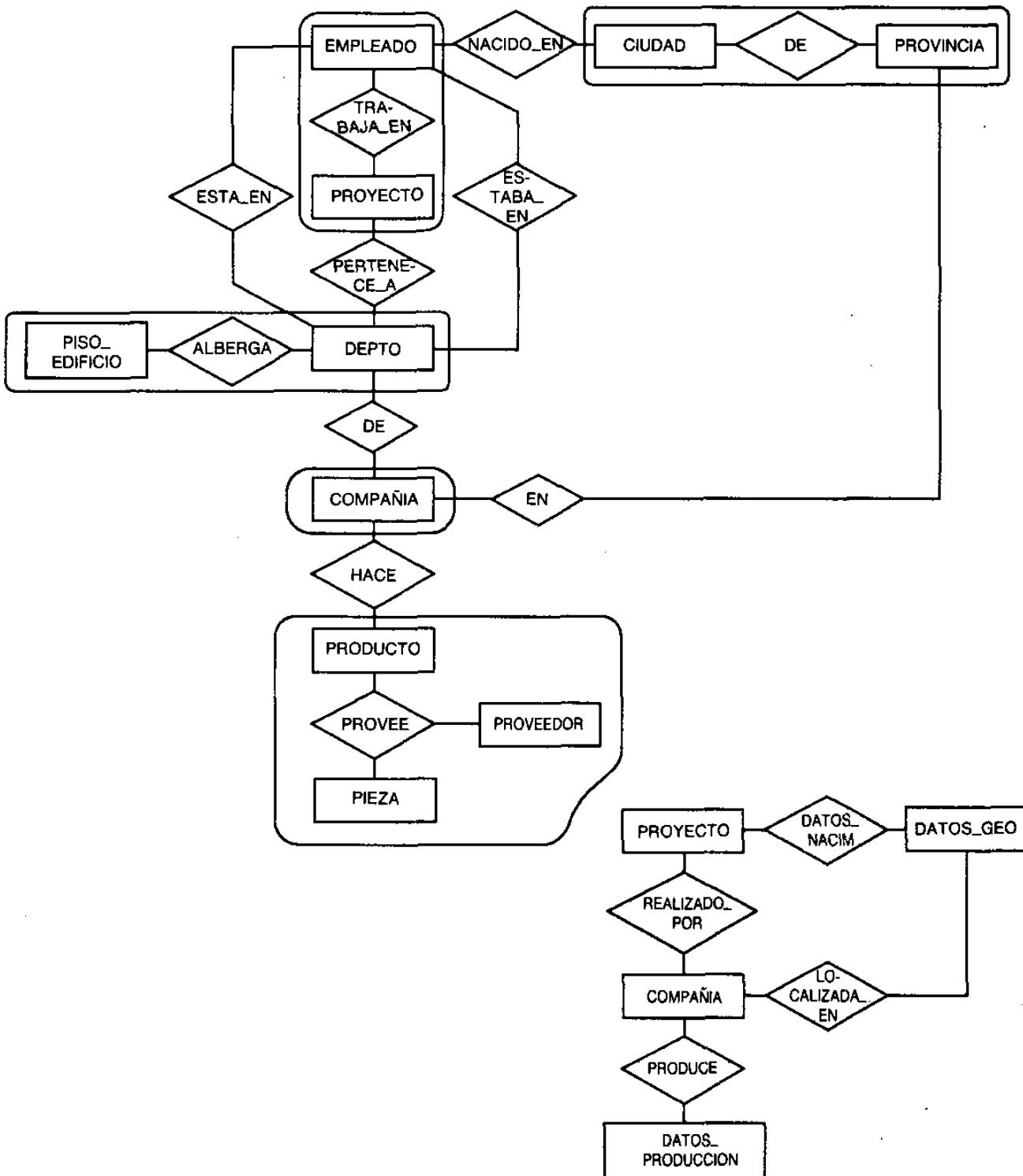
(b) Primera partición

Figura 7.2.Bis Ejemplos de elecciones de particiones (continuación).



(c) Segunda partición

Figura 7.2.2Bis Ejemplos de elecciones de particiones (*continuación*).



(d) Tercera partición

Figura 7.2.3Bis Ejemplos de elecciones de particiones (continuación).

jor desarrollo de los refinamientos. El proceso de partición se aplica al esquema final S_g para determinar el esquema armazón S_0 . Luego, cada concepto de S_0 se refina para generar S_1 y este proceso se repite hasta que finalmente se produce $S_n = S_0$.

Al final de la actividad de documentación, se puede realizar una verificación de la calidad de la documentación global producida. Los planos de refinamiento deben incrementar tanto como sea posible la capacidad del usuario para entender cómo se genera el esquema a partir de los requerimientos. De acuerdo con este punto de vista, se reconocen dos cualidades principales de los refinamientos:

1. El *equilibrado de esquemas* se refiere a los planos de refinamiento como un todo; una colección de planos de refinamiento está equilibrada en esquemas cuando la razón del número de conceptos entre dos refinamientos consecutivos es aproximadamente la misma en toda la documentación.
2. El *equilibrado de conceptos* se refiere a los refinamientos de conceptos individuales; un refinamiento específico del plano S_1 al plano S_2 está equilibrado en conceptos cuando el número de descendientes en S_2 de todos los conceptos de S_1 es aproximadamente el mismo. Se dice que un concepto C_2 de S_2 *desciende* de un concepto C_1 de S_1 cuando C_2 se produce como refinamiento de C_1 .

El equilibrado de esquemas y de conceptos puede representarse gráficamente usando la imagen de un espacio de diseño representado como un cono, como se muestra en la figura 7.3. El equilibrado de esquemas da lugar a planos equidistantes, y el equilibrado de conceptos resulta en planos horizontales. Cuando se descubre un desequilibrio de conceptos y esquemas, una guía simple para lograr el equilibrio consiste en mover primitivas de refinamiento hacia arriba y hacia abajo en los planos hasta que queden correctamente colocadas en el espacio de diseño.

7.1.1. Ejemplo de documentación

Ahora se aplicará la metodología recién descrita a una situación de la vida real: el diseño conceptual de base de datos de un censo. La figura 7.4 muestra el esquema global de la aplicación (se omiten los atributos, y las interrelaciones están sin nombrar por simplicidad). Obsérvese que PERSONA_RESIDENTE y LUGAR están conectados por dos interrelaciones: RESIDE y NACIDO.

Se escogió un primer esquema abstracto de acuerdo con el criterio descrito líneas atrás. Al examinar el esquema global, vemos que una buena partición se da en términos de cuatro entidades que corresponden a los conceptos de alto nivel PERSONA, GRUPO, REFERENCIA_GEOGRAFICA y ALOJAMIENTO (véase el esquema de la figura 7.5.a). El esquema de la figura 7.5b es el esquema armazón S_0 ; las figuras 7.5c y 7.5d muestran dos planos de refinamiento; otro refinamiento aplicado a la figura 7.5d produce el esquema final, S_g (Fig. 7.4).

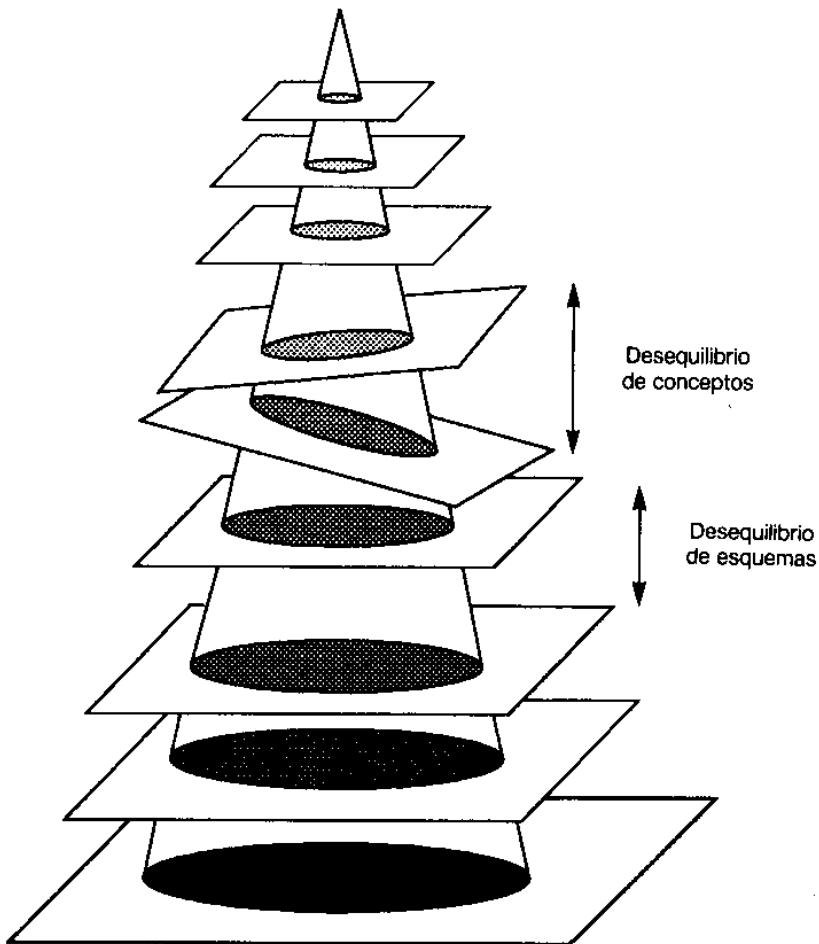


Figura 7.3. Espacio de diseño y planos de refinamiento intermedios.

Obsérvese que en un caso no se aplicó una primitiva descendente pura; las interrelaciones entre CASA_SIN_HABITANTES y las entidades PERSONA_AISLADA_LOCALIZADA_TEMPORALMENTE y SECCION_CENSO se generan durante el refinamiento final con dos transformaciones ascendentes (Fig. 7.4). Se podría usar primitivas descendentes solamente, pero en este caso los refinamientos no hubieran estado tan equilibrados; en particular, algunas interrelaciones a niveles más altos de abstracción hubieran tenido que ser definidas entre ALOJAMIENTO y las entidades PERSONA y REFERENCIA_GEOGRAFICA en el esquema S_0 .

Ahora se puede realizar una verificación de calidad en los planos de refinamiento. Para hacer eso, se sigue la historia de las transformaciones realizadas con cada concepto generado en algún nivel. En los diagramas se muestran los conceptos descendientes por medio de líneas cerradas (Fig. 7.6) Considerese, por ejemplo, la entidad PERSONA, que tiene muchos conceptos descendientes en el esquema final. Como resultado de verificaciones en los distintos planos, se puede concluir que PERSONA está refinada homogéneamente. Lo mismo se aplica a los otros conceptos.

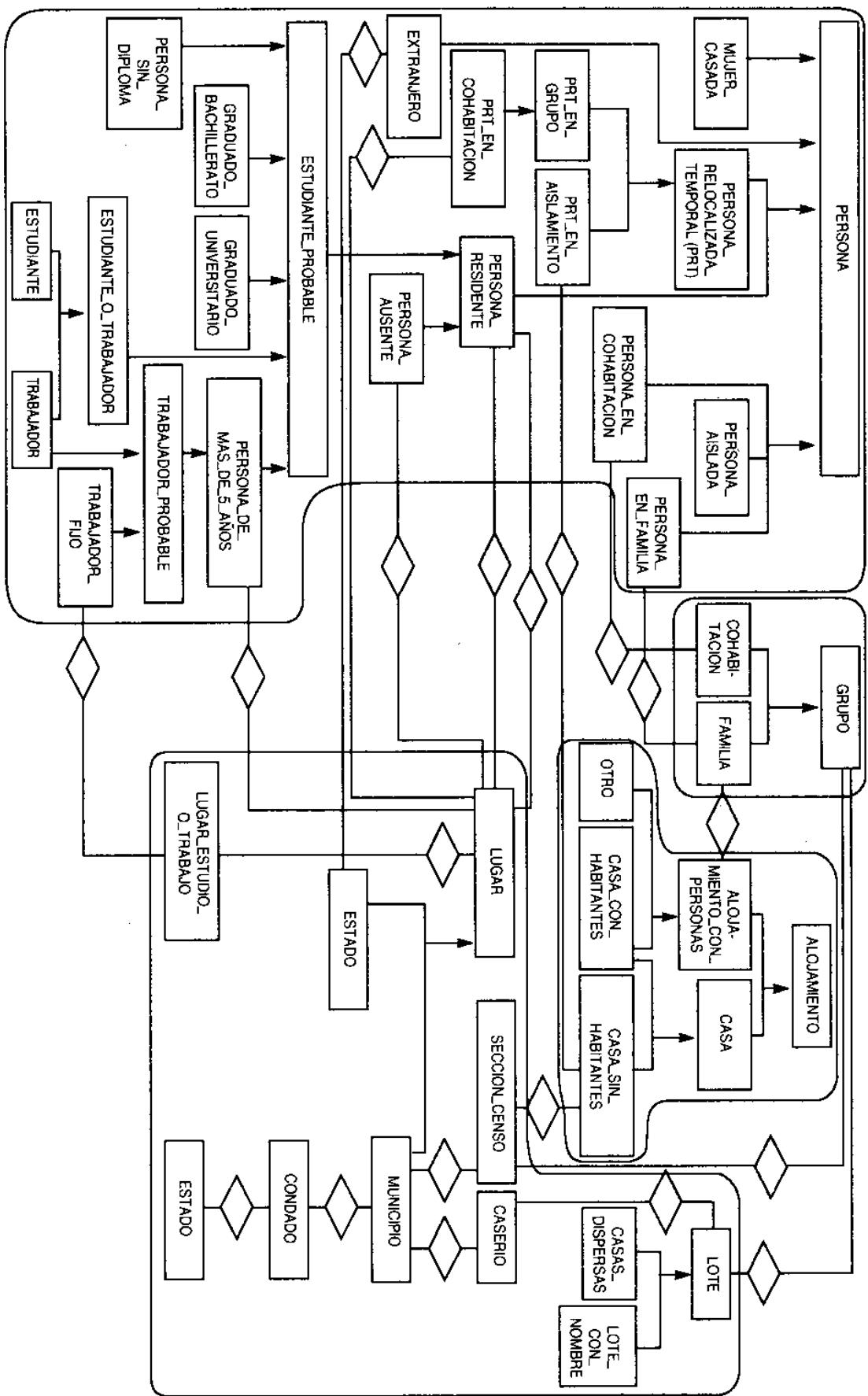
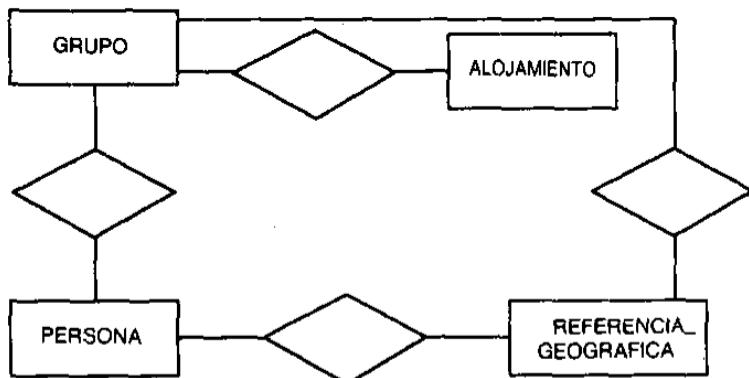
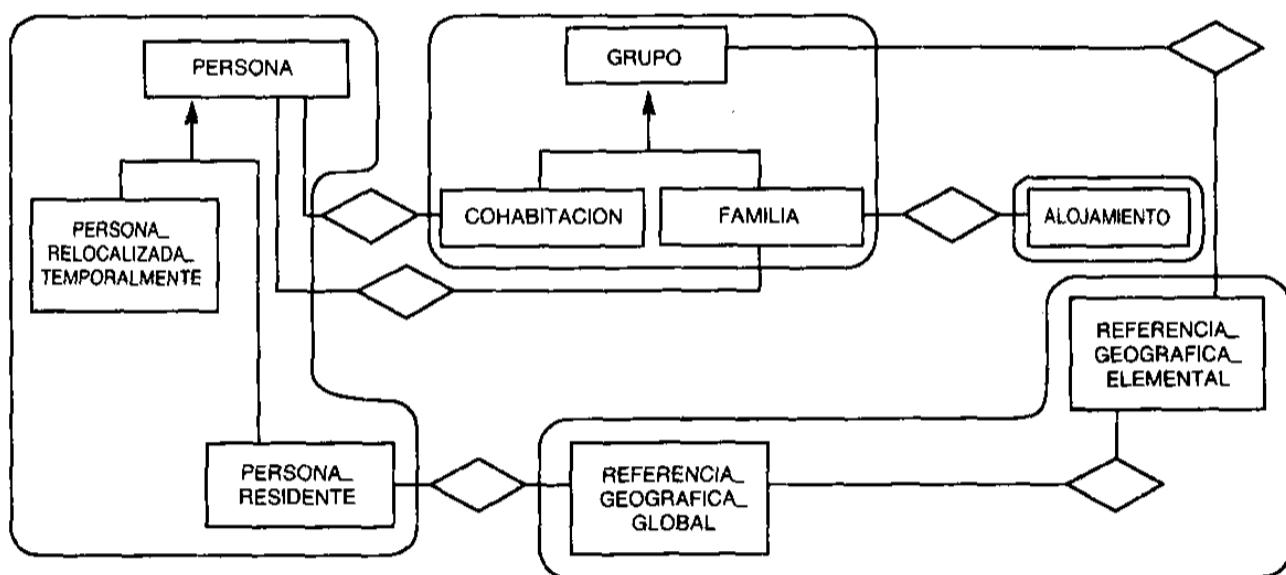


Figura 7.4. Esquema global de un ejemplo de base de datos de un censo.

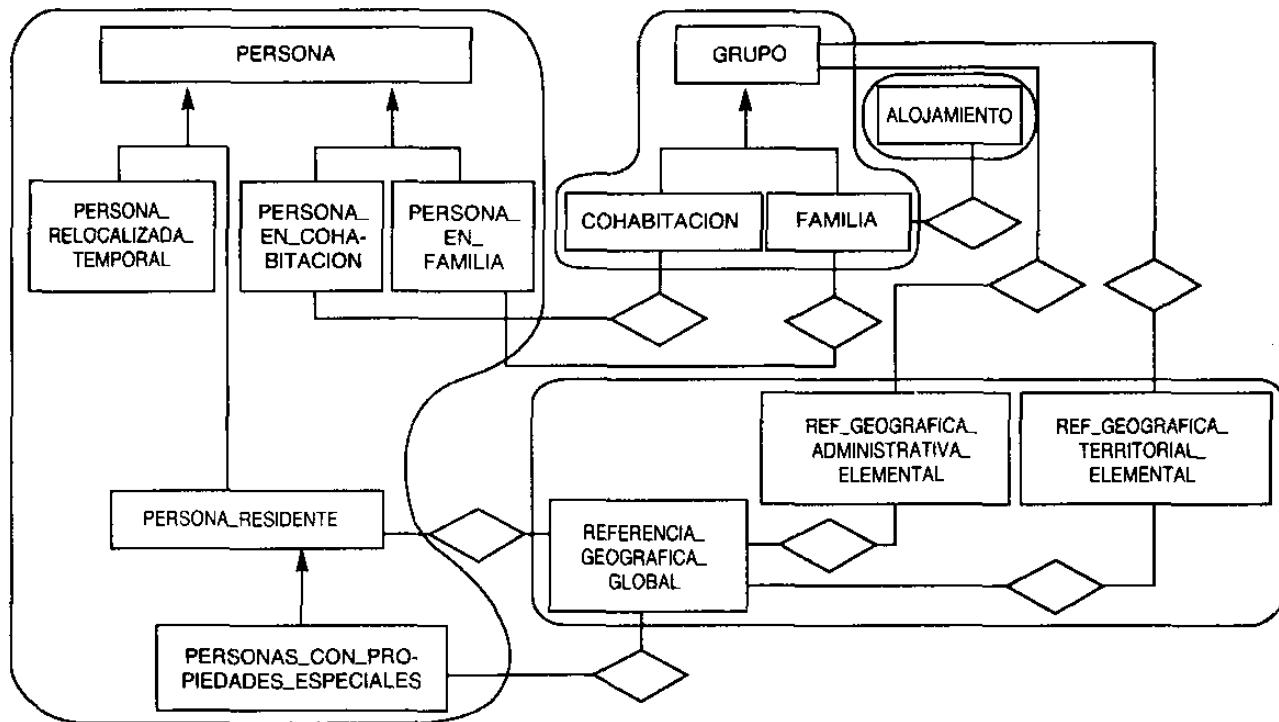


(a) Primer refinamiento

(b) Segundo refinamiento (esquema S_0)**Figura 7.5.** Planos de refinamiento para el ejemplo de base de datos de un censo.

7.2. Mantenimiento de esquemas conceptuales

Cuando los requerimientos cambian, los esquemas conceptuales viejos deben ser actualizados para mantenerlos coherentes con los nuevos requerimientos. Los planos de refinamiento se pueden usar en esta etapa y el nuevo diseño puede ser guiado por transformaciones previas que son ajustadas a los nuevos requerimientos. Pueden presentarse diferentes situaciones, dependiendo del conte-



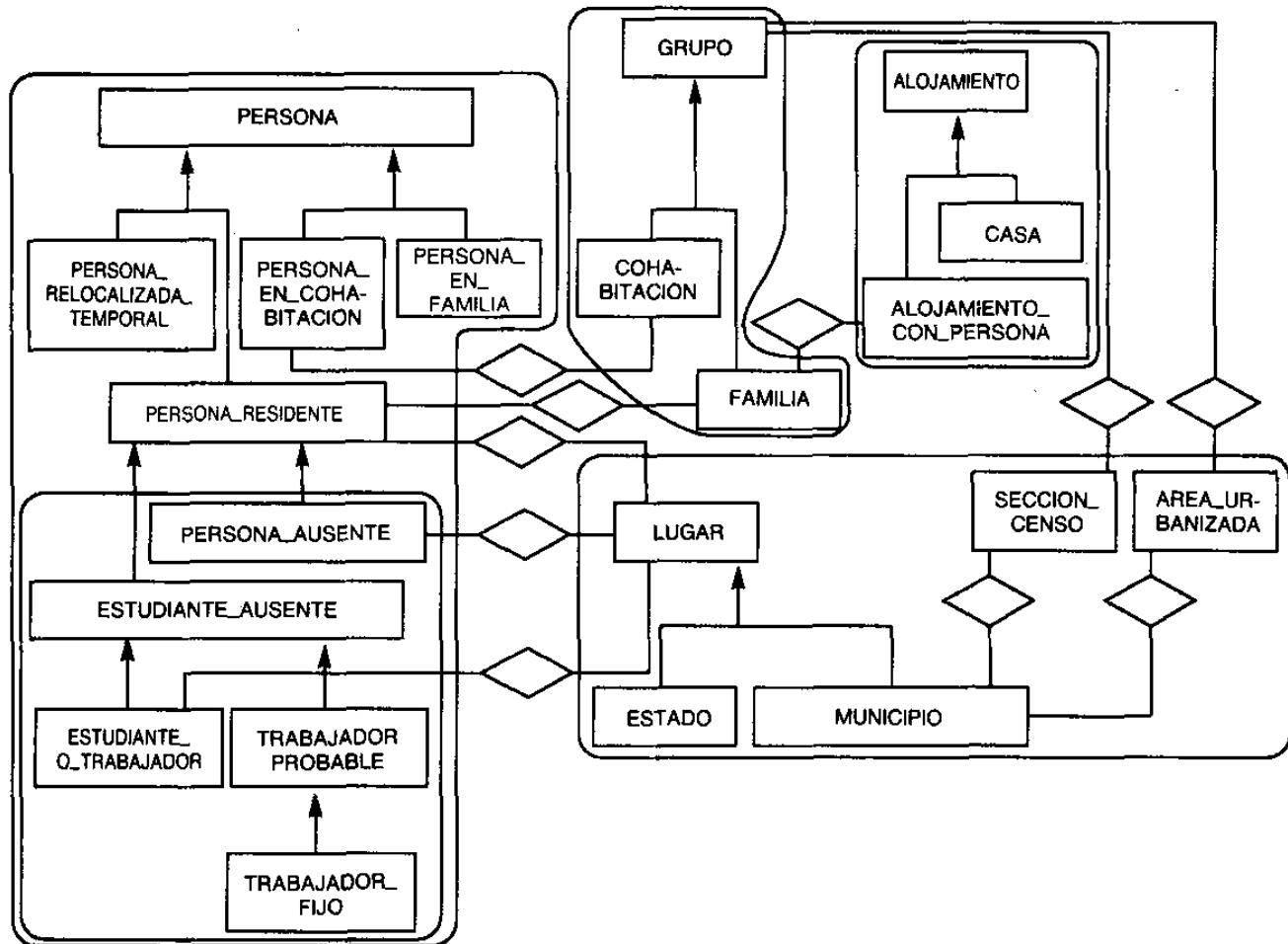
(c) Tercer refinamiento

Figura 7.5.Bis Planos de refinamiento para el ejemplo de base de datos de un censo (*continuación*).

nido de información de los requerimientos viejos y nuevos. Se distinguen cuatro casos:

1. **Superposición alta:** Los requerimientos viejos y nuevos se refieren a la misma aplicación.
2. **Superposición baja:** Los requerimientos se refieren a aplicaciones diferentes pero relacionadas, diseñadas en momentos distintos.
3. **Expansión:** Los nuevos requerimientos contienen por completo a los requerimientos viejos.
4. **Compresión:** Los nuevos requerimientos están contenidos por completo dentro de los requerimientos viejos. Este caso ocurre típicamente cuando el diseño conceptual implica un superconjunto de los requerimientos que serán automatizados.

Es claro que el beneficio de la existencia de documentación del esquema conceptual es mayor en los casos de superposición alta, expansión y compresión; en el caso de superposición baja, suele ser conveniente diseñar el nuevo esquema partiendo de cero.



(d) Cuarto refinamiento

Figura 7.5.2Bis Planos de refinamiento para el ejemplo de base de datos de un censo (continuación).

La idea principal que se presenta en este apartado es que el mantenimiento de esquemas puede estar guiado por planos de refinamiento previos; se distinguen en cada plano las transformaciones que pueden volverse a usar y aquellas que no se pueden aplicar más. Considerense dos planos de refinamiento relacionados S_1 y S_2 , como se muestra en la figura 7.6. En relación a los nuevos requerimientos, las transformaciones pertenecientes al diseño viejo pueden ser clasificadas como sigue:

1. Las *transformaciones sin cambios* pueden aplicarse en el diseño nuevo sin ningún cambio.
2. Las *transformaciones retiradas* no pueden aplicarse más, pues se aplicarían a conceptos que ya están en el nivel de detalle final del esquema nuevo.

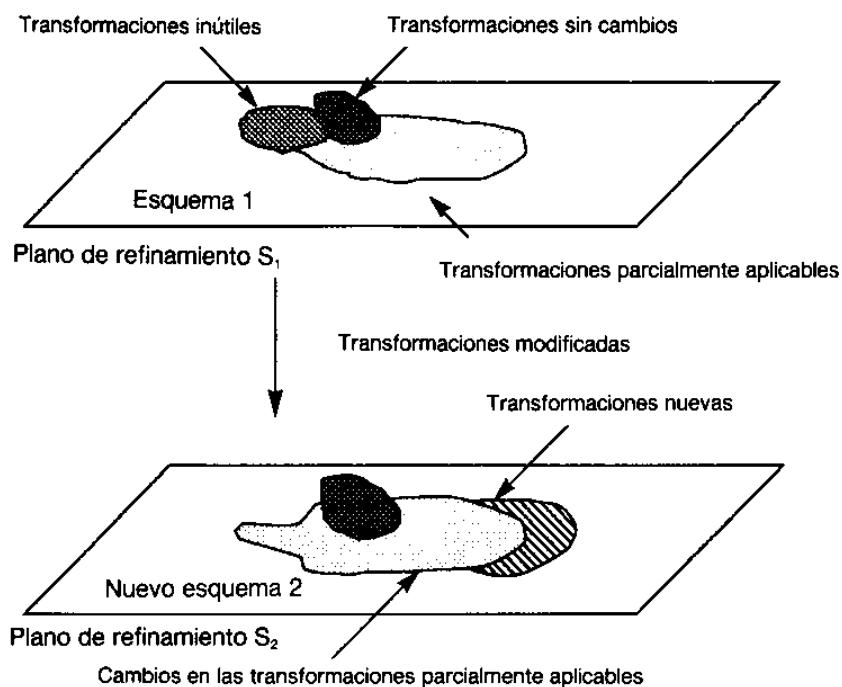


Figura 7.6. Mantenimiento de esquemas aplicado a planos de refinamiento.

3. Las *transformaciones parcialmente aplicables* pueden aplicarse con modificaciones debido a los cambios en los requerimientos.

Además pueden presentarse *transformaciones nuevas*; éstas se aplican a conceptos que previamente estuvieron en el nivel final de refinamiento pero se han convertido en conceptos no terminales en el diseño nuevo.

Una vez clasificadas las transformaciones en estos cuatro grupos, hay que modificar las que son parcialmente aplicables de acuerdo con los nuevos requerimientos y añadir las transformaciones nuevas; de esta manera se obtiene el nuevo esquema S_2 . Aplicando este procedimiento para cada plano de refinamiento, se obtiene finalmente el esquema global nuevo. Como la importancia de los conceptos puede cambiar en el nuevo diseño, se debe verificar el balance de conceptos y esquemas en la documentación del esquema resultante.

Ahora se aplicará la metodología a un ejemplo. Partiendo de la información recolectada en el censo, se concentra la atención en una aplicación nueva, enfocada al estudio de los fenómenos sociales implicados en la migración de familias, personas residentes (específicamente trabajadores), y preferencias políticas de la población en diferentes áreas geográficas. En este caso, el esquema nuevo es significativamente menos detallado que el viejo: éste es un caso de compresión (la mayoría de los conceptos del esquema nuevo están contenidos en el esquema viejo), con una expansión limitada (debido a la existencia de pocos conceptos nuevos). El primer refinamiento permanece sin cambio. Apli-

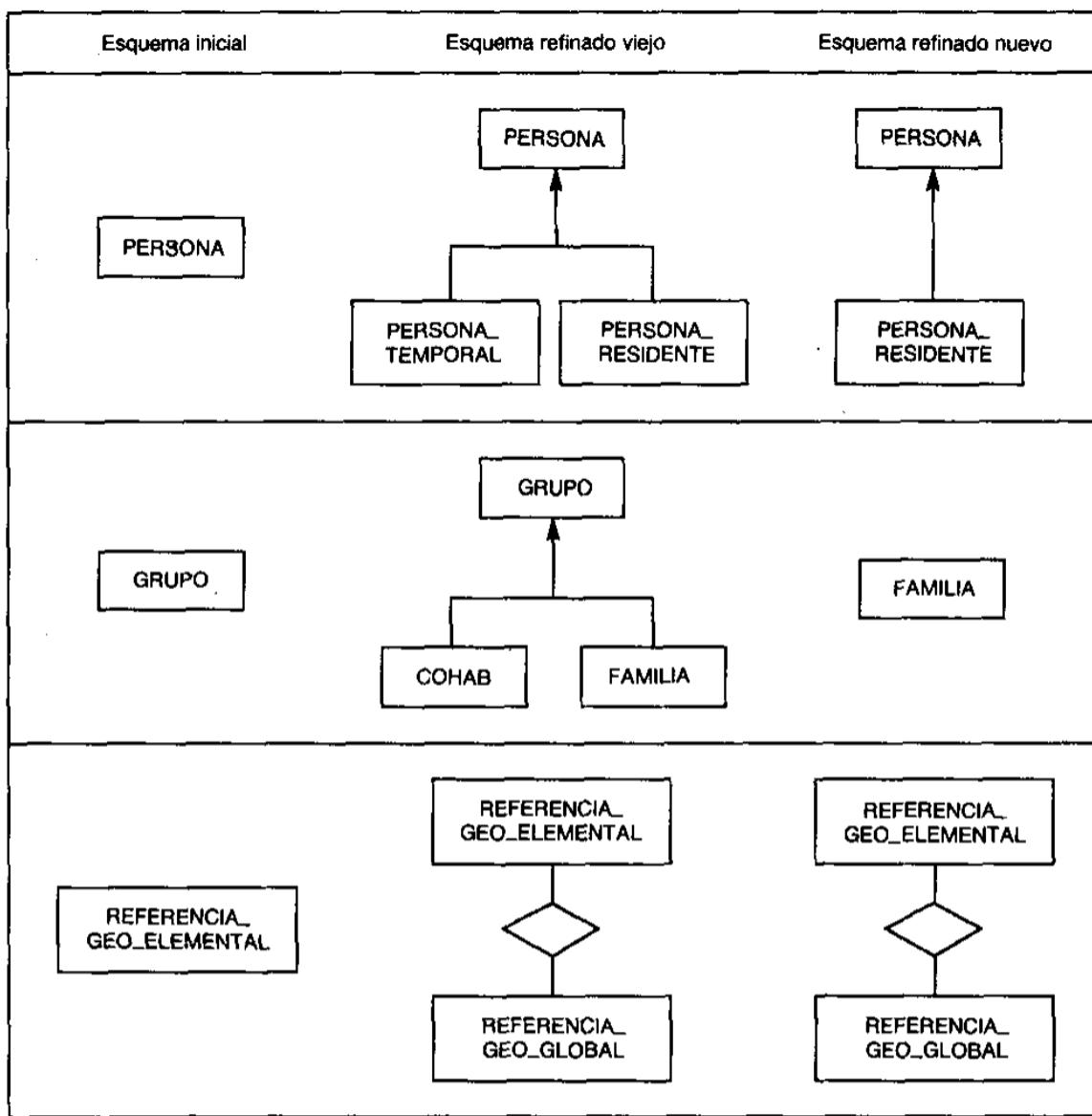


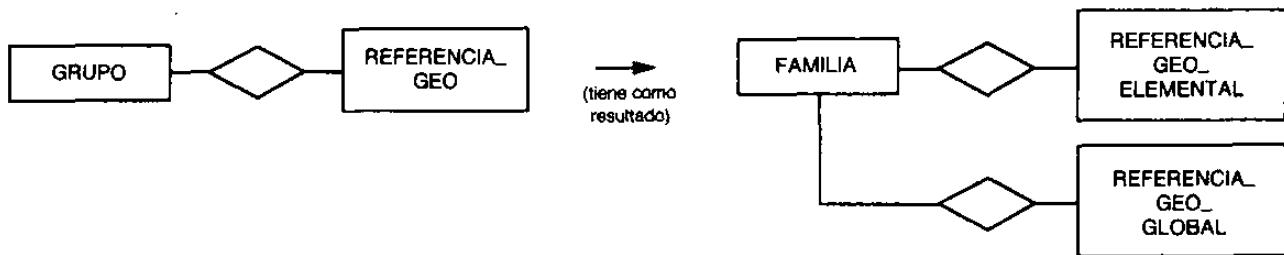
Figura 7.7. Comparación de transformaciones viejas y nuevas para el segundo refinamiento.

cando la metodología de mantenimiento al segundo refinamiento, se descubre que todas las transformaciones son parcialmente aplicables. Las transformaciones nuevas se comparan con las viejas en la figura 7.7. En relación con la herencia de conexiones entre las entidades, se observa que la interrelación entre GRUPO y REFERENCIA_GEO se expande ahora (Fig. 7.8) en dos interrelaciones diferentes entre FAMILIA y REFERENCIA_GEO_ELEMENTAL y REFERENCIA_GEO_GLOBAL, respectivamente. El nuevo segundo refinamiento se muestra en la figura 7.8.

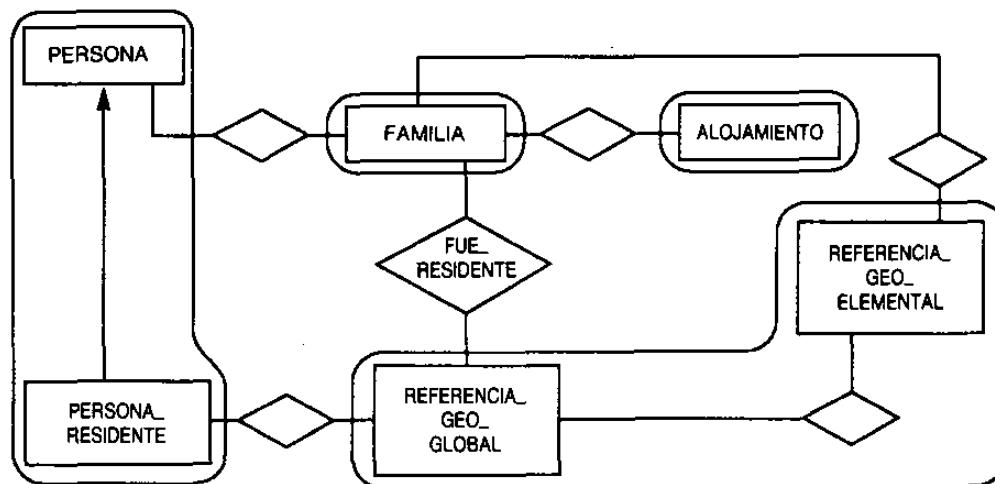
El tercer refinamiento (Fig. 7.9) incluye los dos refinamientos previos, el ter-



(a) Herencia vieja de conexiones



(b) Herencia nueva de conexiones



(c) Nuevo esquema

Figura 7.8. Nuevo segundo refinamiento.

cero y el cuarto. Se observa que varios refinamientos en el diseño viejo, específicamente aquellos que conciernen a las personas, no se aplican en el diseño nuevo. Además, hay que añadir a los descendientes de **REFERENCIA_GEO_GLOBAL** una nueva entidad, **PARTIDO_POLITICO**, que está conectada a **MUNICIPIO** a través de una interrelación generada por una transformación ascendente.

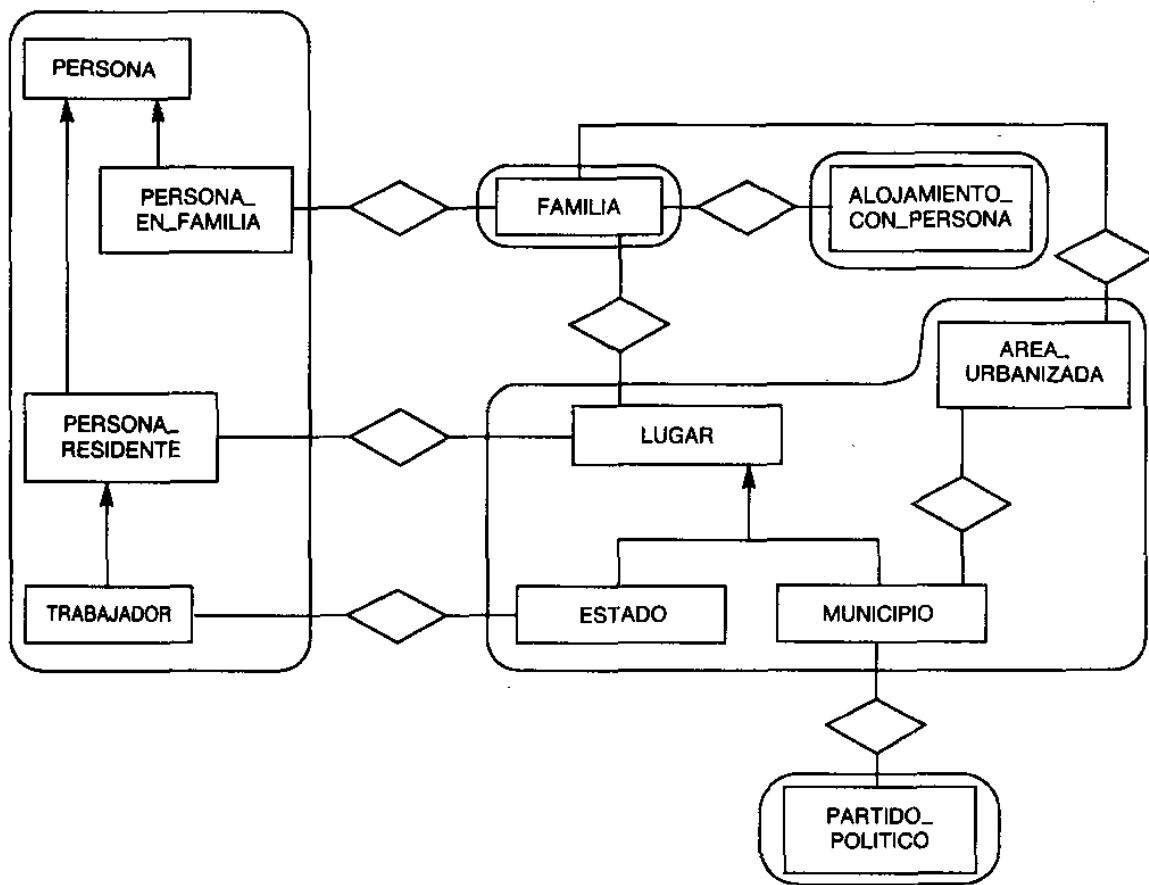


Figura 7.9. Nuevo tercer refinamiento.

Por último, se puede producir directamente un esquema final (Fig. 7.10). De nuevo, no se muestran en detalle todos los refinamientos viejos y nuevos; se observa que todos los refinamientos en los que participa **MUNICIPIO** permanecen sin cambio.

7.3. Estructura y diseño de un diccionario de datos

Los sistemas de información modernos tienden a ser más y más complejos y se caracterizan por varios tipos de heterogeneidad. Por ejemplo, pueden usarse *distintos modelos de DBMS* para representar los datos, como por ejemplo los modelos jerárquico, de redes y relacional. Aparte de las bases de datos, muchos sistemas de software (como las hojas electrónicas de cálculo, bases de datos de multimedios y bases de conocimientos) almacenan otros tipos de datos, cada uno de ellos con su propio modelo de datos. Además, los mismos datos pueden ser vistos por varios usuarios de la base de datos en *diferentes niveles de abstracción*.

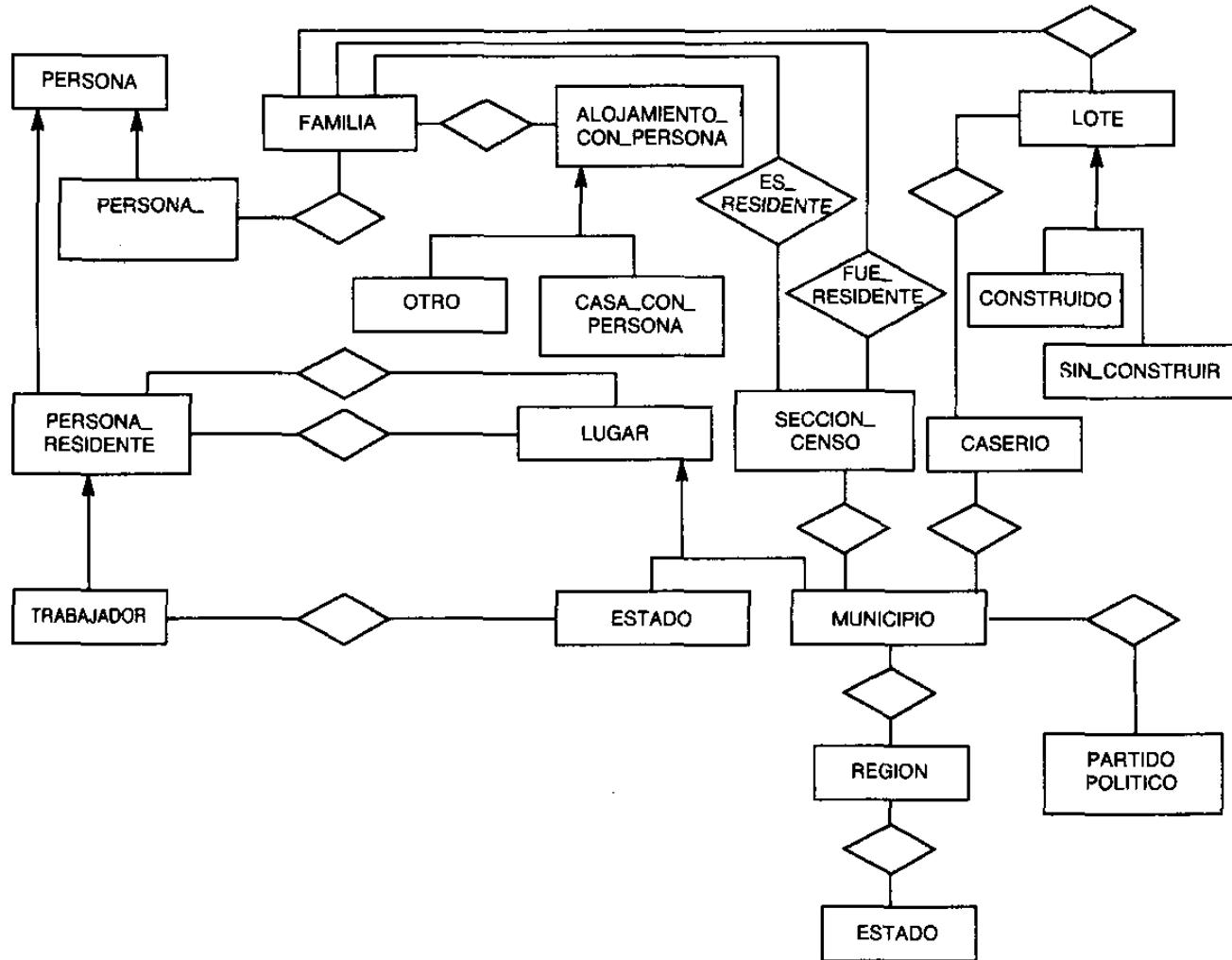


Figura 7.10. Nuevo esquema final.

tracción. Por ejemplo, el presupuesto es visto por los directores en términos del equilibrio estimado y real y por la administración en términos de elementos detallados de entrada/salida.

Debido a tales diferencias, los usuarios consideran un reto entender el significado de todos los tipos de datos que circulan en una organización. Son ayudados por la disponibilidad de un **diccionario de datos**, que es un depósito integrado de todos los tipos de datos producidos, controlados, intercambiados y mantenidos en la organización. Esta información se debe representar de una manera uniforme, ayudando así a los usuarios a entender el significado de los recursos de datos y las formas en que los datos se relacionan. La representación de las relaciones entre los datos y los otros recursos implicados en el sistema de información, como procesos, unidades de organización, personal y tecnologías, es también parte del diccionario de datos.

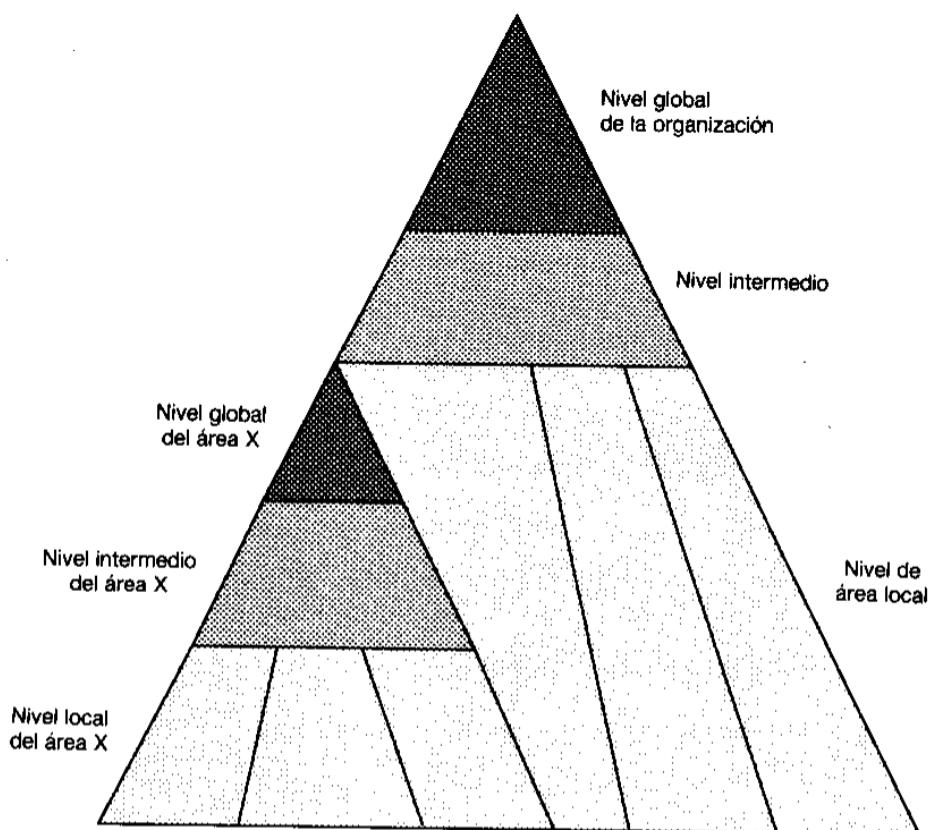


Figura 7.11. Estructura recursiva del diccionario.

Desarrollar un diccionario de datos integrado es un reto importante. En particular, los diccionarios deben ser *activos*, cambiando su contenido conforme nuevas aplicaciones o bases de datos se agregan al sistema de información. Un diccionario activo trabaja con un DBMS, de manera que características como la optimización de consultas dentro del DBMS puedan hacer uso de manera inmediata de la información del diccionario; así pues, construir un diccionario activo de datos es un reto tecnológico. En este capítulo sólo nos interesa el diseño conceptual del diccionario de datos. Este es principalmente el problema de integrar los esquemas conceptuales de las diversas bases de datos descritas en el diccionario de datos; sin embargo, un problema adicional en la integración surge de las diferencias en los niveles de abstracción en los cuales los esquemas conceptuales están descritos.

7.3.1. Estructura de un diccionario de datos

La estructura conceptual de un diccionario de datos consiste en tres capas, como se muestra en la figura 7.11:

1. Una **capa global** (vértice de la pirámide), donde se representan los datos comunes para la organización completa. En esta capa los datos se describen en un nivel muy abstracto, para tener una representación global uniforme (uno o más esquemas armazón globales en diferentes niveles de refinamiento).
2. Una **capa local** (base de la pirámide), donde los datos se representan como parte de sistemas de información y unidades organizacionales específicos. En esta capa, todos los tipos de estructuras son significativos.
3. **Varias capas intermedias**, que establecen la correspondencia entre la capa global y los niveles locales. Las vistas son los mecanismos típicos que se usan en este nivel para relacionar esquemas que se refieren a sistemas de información y unidades organizacionales diferentes.

Estos tres niveles de arquitectura son suficientes para organizaciones pequeñas o medianas; en organizaciones complejas, donde existen varias áreas con estructuras altamente independientes, la arquitectura recién descrita puede repetirse varias veces dentro de las áreas y entre ellas. Esto da lugar a una estructura recursiva para el diccionario, que se ilustra en la figura 7.11.

7.3.2. Construcción y mantenimiento de un diccionario de datos

El diccionario de datos integrado incluye, por supuesto, un amplio número de esquemas no homogéneos; es fundamental darles una estructura. Los esquemas de un diccionario de datos pueden estar relacionados a través de los tres mecanismos básicos de estructuración: refinamientos, vistas e integración. Los **refinamientos** nos permiten modelar el mismo fragmento de la realidad en términos de varias descripciones en diferentes niveles de detalle. El refinamiento es el paradigma típico usado en las metodologías descendentes desarrolladas para el análisis de datos. Estas se definieron en el capítulo 3 y se usan a lo largo de todo este capítulo. Las **vistas** son versiones personalizadas de fragmentos de un esquema. Las vistas pueden servir, por ejemplo, para relacionar un esquema global de una aplicación con los conceptos específicos requeridos por un grupo de usuarios. Por ejemplo, se puede extraer unos cuantos conceptos de un esquema complejo y considerarlos como una vista. La **integración** se aplica a dos o más esquemas para obtener una visión global de los datos administrados por un sistema de información completo, un área de organización o un sistema distribuido. El proceso se analiza en el capítulo 5.

Ahora se aborda el problema de construir un diccionario de datos sobre la base de estos métodos. La estructura de alto nivel de la metodología se muestra en la figura 7.12. Se supone como entrada a la actividad de diseño un conjunto de esquemas, cada uno asociado con una aplicación específica y documentado en términos de un conjunto de planos de refinamiento. Obsérvese que algunos de los esquemas que serán integrados pueden ser esquemas lógicos. En ese caso se debería aplicar retroingeniería al esquema lógico para reconstruir el esquema conceptual a partir del esquema lógico (este problema se trata en la tercera parte

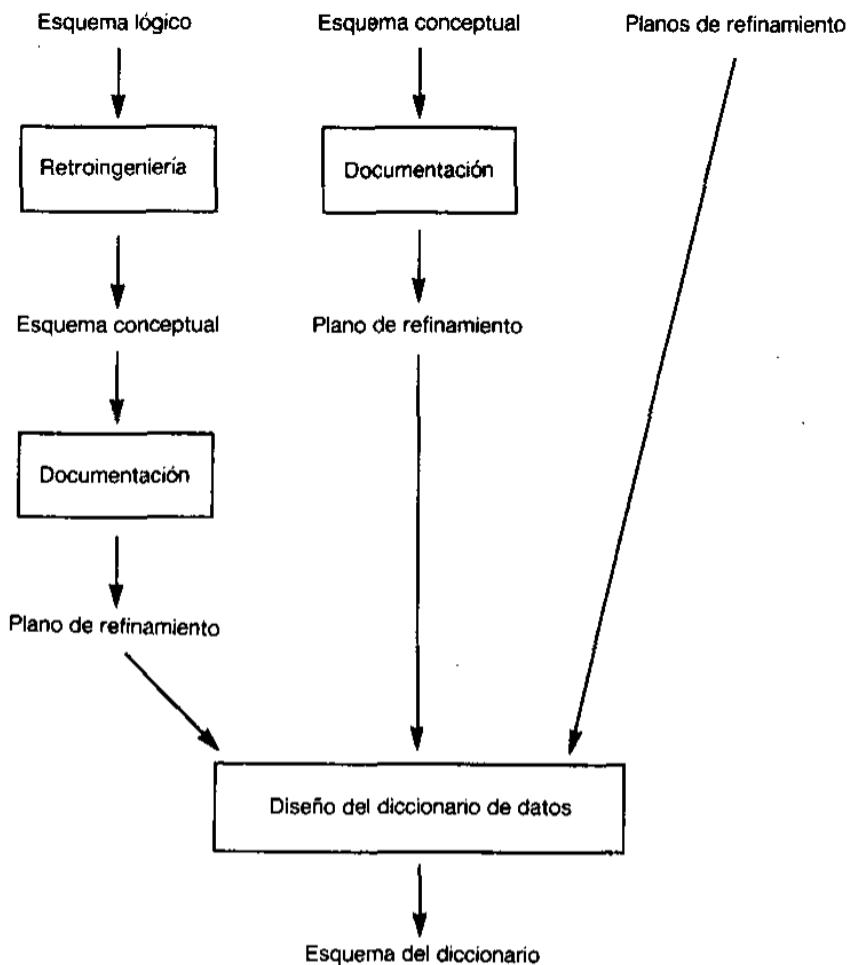


Figura 7.12. Entradas para el diseño de un diccionario de datos.

de este libro, apartado 12.5). Después de producir un esquema conceptual, se debe documentar como se describe en el apartado 7.1.

La principal dificultad en el diseño de un diccionario de datos surge de la falta de homogeneidad en la entrada. Incluso si los esquemas están documentados en forma descendente, los planos de refinamiento pueden no ser fácilmente comparables uno con el otro; cierta documentación puede incluir varios planos y ser muy detallada, con buen equilibrio de esquemas y conceptos; otra documentación, por el contrario, puede estar simplemente en borrador, sin equilibrio. Así pues, se necesita establecer las correspondencias entre ellas en diferentes niveles de abstracción e integrar esquemas en niveles comparables. Aquí se esboza una metodología conceptual para producir la integración.

El primer paso consiste en colocar los esquemas correspondientes a diferentes aplicaciones en una rejilla. Cada columna de la rejilla (Fig. 7.13) representa una aplicación o unidad funcional de la organización, y cada fila representa un nivel de abstracción. Se determina primero una representación integrada del es-

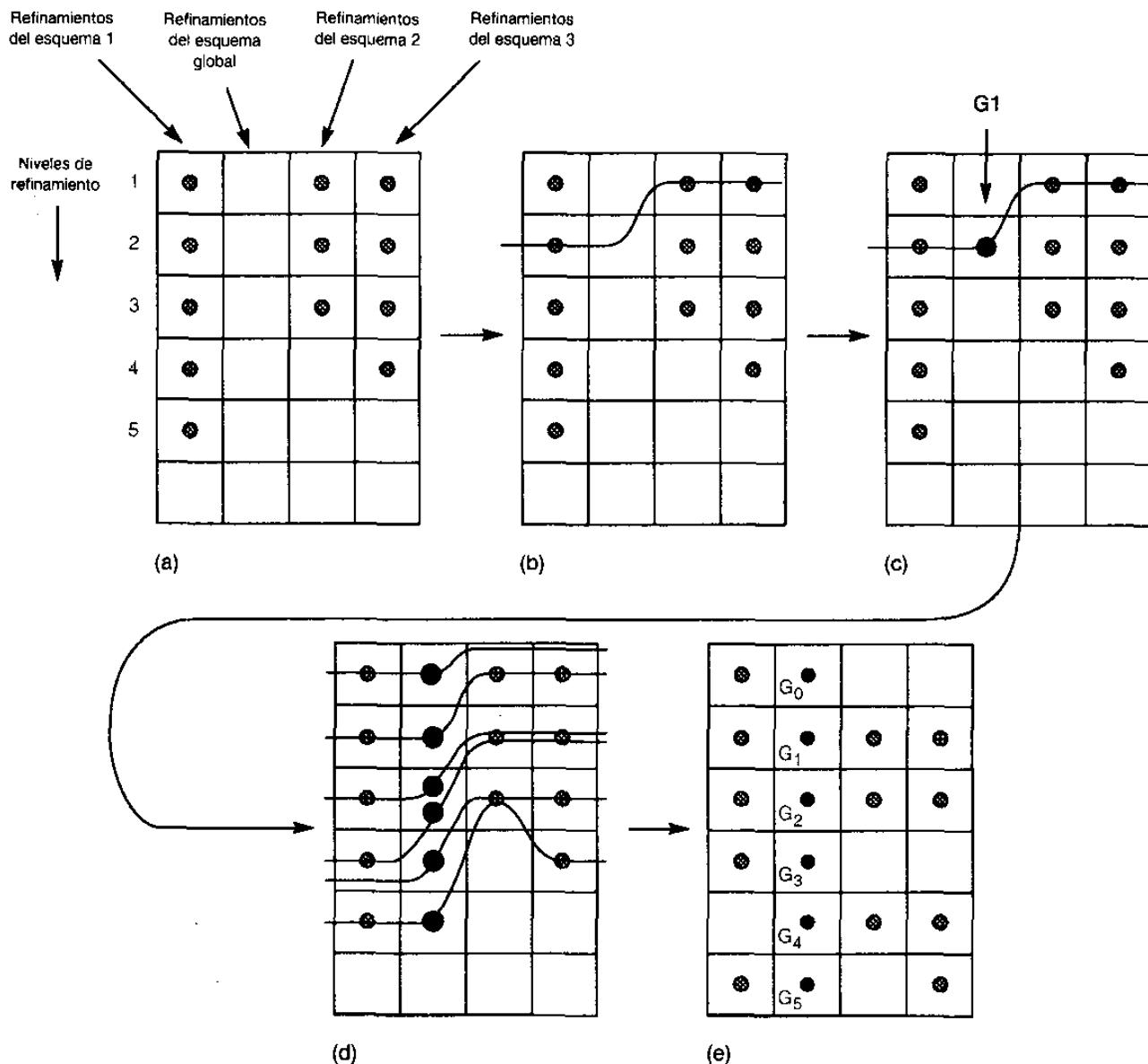


Figura 7.13. Visualización de los pasos para construir un diccionario de datos integrado.

esquema global; luego se produce el conjunto completo de refinamientos para el esquema global, reestructurando posiblemente esquemas locales. Se tratarán los pasos específicos con más detalle:

1. *Seleccionar el primer conjunto de esquemas locales a ser integrado.* Se selecciona un conjunto de planos de refinamiento candidatos, que se vean adecuados para la integración. La idea aquí es que, debido a los diferentes ta-

maños de los sistemas de información que operan en la organización, se debe escoger el nivel de abstracción en el cual los conceptos sean más fáciles de fusionar, y luego se determina para cada esquema local el plano de refinamiento que corresponde a ese nivel de abstracción.

2. *Integrar esquemas locales para producir un primer prototipo del esquema global.* Los esquemas seleccionados en el paso 1 son luego integrados. Los esquemas locales pueden incluir conflictos producidos por sinónimos, homónimos o diferentes tipos para los mismos conceptos. Los conflictos deben descubrirse y resolverse usando la metodología del capítulo 5.
3. *Producir el conjunto completo de planos de refinamiento del esquema global.* Una vez que un esquema global ha sido producido para un nivel de abstracción particular, todos los esquemas locales en diferentes niveles de abstracción deben integrarse también, con el fin de producir una visión del esquema global a través de los planos de refinamiento. Esto requiere no sólo moverse *debajo* del esquema integrado, sino también *por encima*, dando descripciones más abstractas del diccionario de datos.

La producción de planos de refinamiento para el esquema global puede no ser factible para organizaciones grandes. Cuando el esquema global tienda a ser demasiado grande, se puede suspender la generación de planos de refinamiento globales y proceder recursivamente a producir diccionarios locales. Por otro lado, puede suceder que los planos de refinamiento de alto nivel usen conceptos muy abstractos que sean, en consecuencia, no significativos. En este caso no tiene sentido forzar la producción de tales niveles y es mejor cortar la pirámide, dejando los esquemas locales de alto nivel sin integrar.

Un ejemplo de integración de diccionario de datos desarrollado por las tres etapas expuestas anteriormente se muestra en la figura 7.13. Inicialmente, están presentes tres esquemas; el primero tiene cinco planos de refinamiento, el segundo tiene tres y el tercero tiene cuatro. Cada plano de refinamiento se representa como un punto en la matriz de la figura 7.13a; una columna de la matriz se deja vacía y se llenará con esquemas del diccionario de datos global. La figura 7.13b muestra que se selecciona un plano de refinamiento para cada esquema; los tres esquemas representados de esta manera constituyen el primer conjunto de esquemas locales por integrar. Su integración produce el esquema global G_1 , que se muestra en la figura 7.13c. Obsérvese que G_1 se obtiene integrando un esquema (esquema 1) en el segundo nivel de refinamiento y dos esquemas (a saber, esquema 2 y esquema 3) en el primer nivel de refinamiento. En seguida se determina un conjunto completo de planos de refinamiento globales (G_0 a G_5), como se ilustra en la figura 7.13d. G_0 es más abstracto que G_1 ; G_2 a G_5 representan refinamientos subsecuentes de G_1 . Finalmente, la figura 7.13e muestra cómo los planos de refinamiento originales se colocan en correspondencia con los esquemas globales.

7.4. Resumen

Este capítulo concluye el análisis de metodologías para el diseño conceptual, dando orientaciones generales para tres actividades importantes en el ciclo de vida del diseño conceptual: documentar una sesión de diseño, mantener un esquema conceptual cuando cambian los requerimientos y construir un diccionario de datos integrado.

Ejercicios

- 7.1. Considere los esquemas producidos para los ejercicios de los capítulos 3 y 4. Produzca una documentación para cada uno de ellos, incluyendo un conjunto de planos de refinamiento.
- 7.2. Considere de nuevo el esquema producido para el ejercicio 4.3. Tenga en cuenta que los siguientes requerimientos han cambiado:
 - a) Las agencias pueden ahora reservar bloques de asientos; en este caso, sólo se almacena el número de asientos reservados. Las agencias también pueden conseguir descuentos sobre el precio de los billetes, dependiendo del volumen de billetes comprados en los últimos tres años.
 - b) Se pueden organizar ocasionalmente viajes especiales para eventos específicos. Para esos viajes se desea incluir en la base de datos una descripción del evento, pero no se necesita representar las paradas intermedias, porque no las hay.
- 7.3. Considere el esquema final obtenido para el ejercicio 3.4 y la correspondiente documentación descendente. Suponga los siguientes *cambios en los requerimientos* en términos de añadir datos políticos:
 - a) Represente los datos acerca del tipo de escuela donde los participantes obtuvieron su último grado y la ciudad donde está la escuela.
 - b) Para cada estado y cada partido político, represente el porcentaje del partido en las últimas elecciones políticas, con delegados (nombres y edad); para cada partido, represente el porcentaje nacional.
 - c) Para cada estado y cada partido político, represente el porcentaje del partido en todas las elecciones en la historia del país.

Realice el mantenimiento que se requiera por los cambios en los requerimientos recién citados, usando la metodología descrita en el apartado 6.3.

- 7.4. Partiendo del esquema de la figura 4.8, construya una documentación conceptual, escogiendo el número y tipo de planos de refinamiento y mostrando las primitivas usadas en los refinamientos. A continuación, use pla-

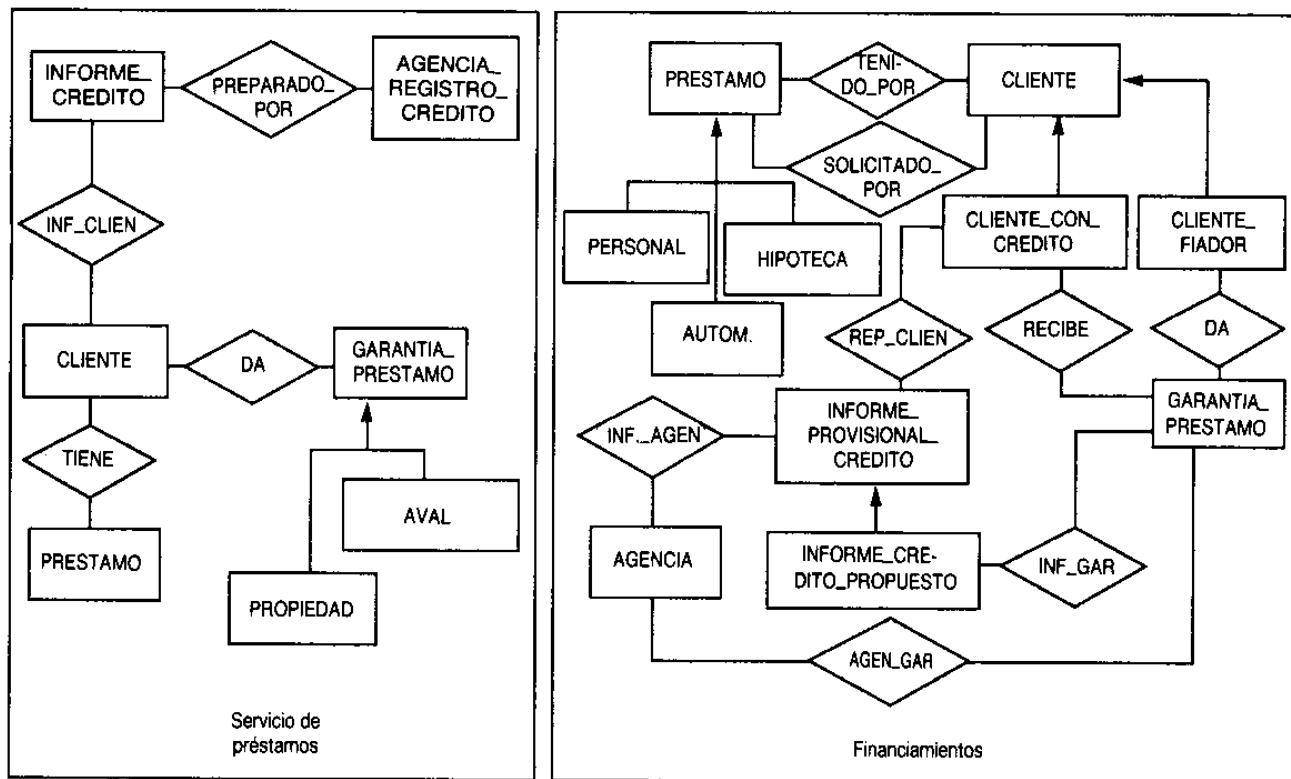
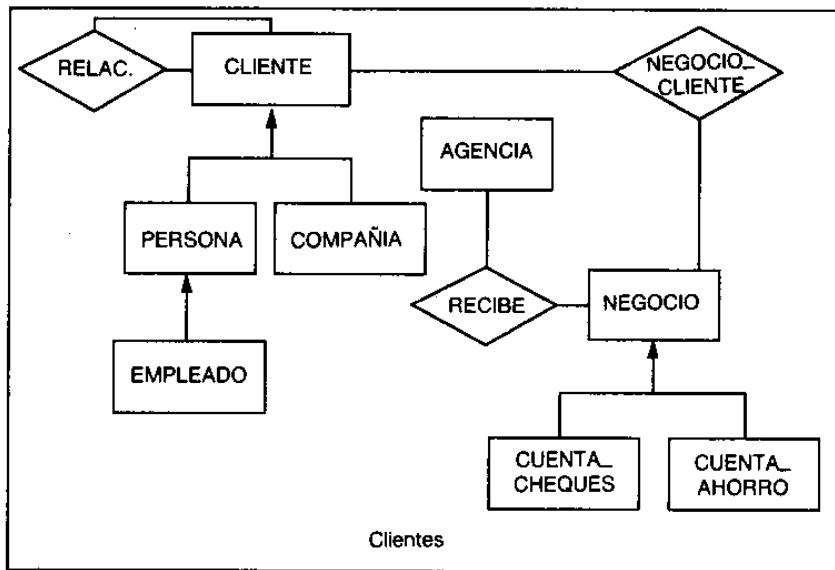


Figura 7.14. Tres esquemas para el sistema de información de un banco.

nos de refinamiento de acuerdo con la metodología de mantenimiento del apartado 7.2 para cubrir los siguientes requerimientos:

- a) El esquema de la figura 4.8 fue concebido para una universidad específica de un estado; se quiere ahora representar los mismos requerimientos para todas las universidades de un estado dado.
 - b) Para cada curso, almacene información adicional sobre las aulas en las que se imparte el curso, y el horario correspondiente.
 - c) Ya no interesa la ciudad de nacimiento de los estudiantes en general, sólo de los estudiantes graduados.
 - d) Para cada profesor, interesan las publicaciones y subvenciones que él o ella administra (seleccione los atributos convenientes).
- 7.5. Haga este experimento: Partiendo del esquema del científico y del esquema del bibliotecario que se usaron en el capítulo 5 para la metodología de integración, produzca para cada uno de ellos un conjunto de planos de refinamiento. Haga lo mismo para el esquema global. Intenta ahora integrar los refinamientos producidos para los esquemas del científico y del bibliotecario y compárelos con los refinamientos producidos para el esquema integrado. Compare los dos conjuntos de refinamientos y comente las posibles diferencias.
- 7.6. Produzca un esquema conceptual de la estructura del diccionario de datos, representando toda la metainformación que pueda ser útil en su administración.
- 7.7. Desarrolle una metodología para el mantenimiento de esquemas que tenga en cuenta la disponibilidad de un diccionario de datos durante el mantenimiento.
- 7.8. Construya un diccionario de datos pequeño para una aplicación bancaria. Se supone que el banco está organizado en términos de un conjunto de divisiones. Se tiene interés en la división de préstamos, que puede considerarse como compuesta por tres áreas funcionales: clientes, financiamiento y servicios de préstamos.
La oficina de clientes emplea información personal de los clientes. La de financiamiento se encarga de la evaluación de los préstamos nuevos. El departamento de servicios de préstamos resuelve los problemas relacionados con los préstamos, juicios hipotecarios, refinanciamientos, etc. Los tres esquemas de datos correspondientes se muestran en la figura 7.14.
Usando las metodologías descritas en los apartados 7.1 y 7.3, construya el diccionario de datos correspondiente.

Bibliografía

G.J. Myers, *Composite/Structured Design*, Van Nostrand Reinhold, 1978.

Este libro describe en detalle los conceptos de modularidad, desarrollados en el contexto del diseño de funciones, que se adaptaron en este capítulo al diseño de datos.

C. Batini y G. di Batista, «A Methodology for Conceptual Documentation and Maintenance», *Information Systems*, 13, 1988.

Este artículo describe el enfoque descendente para la documentación y el mantenimiento expuesto en este capítulo.

S. B. Navathe y L. Kershberg, «Role of Data Dictionaries in Database Design», *Information and Management*, 10, núm. 1, enero de 1986, 21-48.

Este artículo trata el uso de diccionarios de datos integrados en el diseño de bases de datos.

G. di Batista; H. Kangassalo, y R. Tamassia, «Definition Librarires for Conceptual Modeling». En C. Batini, ed., *Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

Este artículo expone el concepto de biblioteca de definición, que es similar, en principio, al concepto de diccionario de datos expuesto en este capítulo. Los esquemas de la biblioteca son patrones estructurales vacíos usados para dirigir el proceso de modelado (similares a nuestras primitivas de refinamiento) o bien esquemas conceptuales usados como componentes de esquemas conceptuales más grandes.

C. Batini, G. di Batista, y G. Santucci, «A Methodology for the Design of Data Dictionaries». En *Proc. International Phoenix Conference on Computers and Communications*, Phoenix, 1990.

La metodología para el diseño de diccionarios de datos presentada en el apartado 7.5 es tratada en este artículo que presenta un estudio de caso detallado.

J. Winkler. «The Entity-Relationship Approach and the Information Resource Dictionary System Standard». En C. Batini, ed., *Proc. Seventh International Conference on Entity-Relationship Approach*, Roma, North-Holland, 1988.

Este artículo trata la estructura del sistema de diccionario de recursos de información (IRDS). El IRDS pretende ser la principal herramienta para controlar los activos de recursos de información de la organización. Se propone una arquitectura por capas para el IRDS; las capas, desde la base hasta el vértice de la jerarquía, representan casos del mundo real, los esquemas de datos, la estructura del diccionario y, finalmente, la descripción del modelo (el modelo ER) usado para representar la segunda y tercera capas.

D. Dolk, «Model Management and Structured Modeling: The Role of an Information Resource Dictionary System», *Communications of the ACM*, 31, núm. 6, 1988, 704-18.

La función de un IRDS se extiende en este artículo para captar la técnica denominada *modelado estructurado*, cuyo objetivo es representar en un entorno integrado los esquemas y modelos usados en un amplio espectro de aplicaciones, como ciencias de administración, investigación de operaciones, ingeniería de software y sistemas de bases de datos.

S. Huffman y R. Zoeller, «A Rule-Based System Tool for Automated ER Model Clustering». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

T. Teorey, G. Wei, D. Bolton y J. Koenig, «ER Model Clustering as an Aid for User Communication and Documentation in Database Design». *Communications of the ACM*, 32, núm. 8, 1989, 975-87.

Estos artículos exponen dos enfoques del problema de agrupar objetos ER que se pre-

sentó en la metodología para la documentación de esquemas. Se definen varias operaciones de agrupamiento y tipos de cohesión entre conceptos; pueden ser aplicados repetidamente o usados en una variedad de combinaciones para producir conceptos de más alto nivel.

C.R. Carlson, W. Ji, y A. K. Arora, «The Nested Entity-Relationship Model». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.

Este artículo presenta modelado de niveles múltiples con anidamientos de diagramas ER.

P. Freedman y D. Miller, «Entity Model Clustering: Structuring a Data Model by Abstraction», *The computer Journal*, 29, núm. 4, 1986, 348-60.

Este artículo presenta un enfoque para estructurar «modelos de entidades» dentro de organizaciones. Expone una técnica de agrupamiento para estos modelos y puntualiza ventajas para el departamento de información de la organización y para la computación orientada al usuario final.

Segunda parte

Análisis funcional en el diseño de bases de datos

El término *análisis funcional* indica el modelado de actividades de trabajo dentro de una empresa; una *función* es simplemente una porción de la empresa. El análisis funcional se centra en el entendimiento de cómo cada función usa la información y cómo es intercambiada ésta entre las funciones. El análisis funcional es el primer paso hacia la especificación y diseño de programas de aplicación que operan sobre la base de datos. Este es un esfuerzo importante que debe realizarse en conjunto con el diseño del esquema de base de datos, o inmediatamente después, usando las metodologías clásicas de la ingeniería de software.

La especificación de funciones y flujos de información es muy útil para el diseño conceptual de bases de datos; permite verificar la compleción de la base de datos, esto es, comprobar que todos los datos que requieren las funciones están incluidos en la base de datos y todas las operaciones que manipulan la base de datos son realizadas por algunas funciones. Además, el análisis funcional ayuda a desarrollar una visión procedural de la manera como se usa la base de datos; esta visión se vuelve particularmente importante en la correspondencia subsecuente del esquema conceptual al esquema procesable por DBMS (sistema de gestión de bases de datos).

En esta parte del libro, se propone un enfoque integrado del diseño de datos y funciones en sistemas de información, en el que el análisis funcional apoya e influye sobre el diseño de bases de datos. Se empieza por considerar el análisis funcional por sí mismo; se muestra el *modelo de flujo de datos* para el análisis funcional y una metodología para diseñar esquemas de funciones usando el modelo de flujo de datos. Luego se muestra cómo se puede diseñar

conjuntamente los datos y las funciones. Finalmente, se muestra cómo hacer una especificación funcional de aplicaciones de bases de datos en un nivel muy alto de abstracción por medio de esquemas de *navegación*. Esta parte concluye con un amplio estudio de un caso, que resume todas las técnicas para el diseño conceptual que se han analizado y también se usa como ejemplo en toda la tercera parte.

Análisis funcional usando el modelo de flujo de datos

El análisis funcional se ocupa del modelado de un sistema de información en términos de actividades o procesos, y flujos de información entre ellos. El resultado último del análisis funcional es un **esquema funcional**, que contiene una representación de actividades, flujos de información y otros rasgos. El esquema funcional incluye la representación de aplicaciones de bases de datos y las interacciones entre ellas. Se puede considerar esta representación como el **esquema de procesamiento** de la base de datos, en contraposición al esquema estático de la base datos que ofrece el esquema conceptual de la misma.

Este capítulo presenta las nociones de análisis funcional que son útiles en el diseño de bases de datos. Se usa la misma progresión y organización de argumentos que para el modelado conceptual de datos. Primero se introduce un modelo para el análisis funcional; luego se introducen progresivamente transformaciones primitivas para la construcción de esquemas funcionales, estrategias para dominar el proceso de diseño, y técnicas para el diseño y mantenimiento de esquemas funcionales. Finalmente, se comentan las cualidades de un buen esquema funcional. Este tratamiento se mantiene relativamente simple de manera intencionada; el capítulo no pretende ilustrar todos los problemas del análisis funcional, pero sí presentar los fundamentos autónomos para el capítulo siguiente sobre diseño conjunto de datos y funciones. Una omisión notable es que el capítulo no investiga cómo pueden captarse los requerimientos de procesamiento observando diversas fuentes de entrada.

Los diferentes modelos del diseño funcional destacan diferentes rasgos; por ejemplo, algunos modelos se concentran en los datos y flujos intercambiados entre las actividades, y otros se concentran en la sincronización de actividades definiendo sus condiciones previas y posteriores. Los primeros se aplican generalmente al diseño de sistemas convencionales de procesamiento de datos; los segundos son más apropiados para el modelado de actividades dependientes del

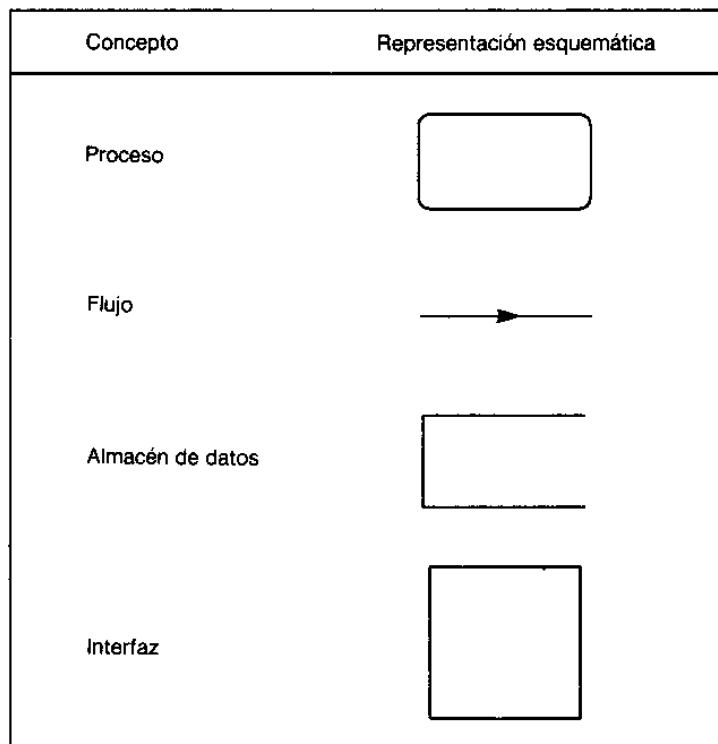


Figura 8.1. Representación esquemática de conceptos de diagramas de flujos de datos (DFD).

tiempo, como los sistemas de tiempo real. En este capítulo se ha escogido un modelo simple, el de flujo de datos, que es muy utilizado para el diseño de sistemas de procesamiento de datos; se examinarán brevemente otros modelos en el apéndice de este capítulo.

8.1. El modelo de flujo de datos para el análisis funcional

El modelo de flujo de datos sustenta los conceptos de proceso, flujo de datos, almacén de datos e interfaz. En la notación de la figura 8.1 los procesos están representados por rectángulos con esquinas redondeadas, los flujos de datos, por líneas dirigidas (flechas), los almacenes de datos, por rectángulos de tres lados, y las interfaces, por rectángulos con esquinas de ángulo recto. Usando estos conceptos, el diseñador construye un esquema funcional, también denominado **diagrama de flujo de datos** (DFD). Los términos de la notación se definen como sigue:

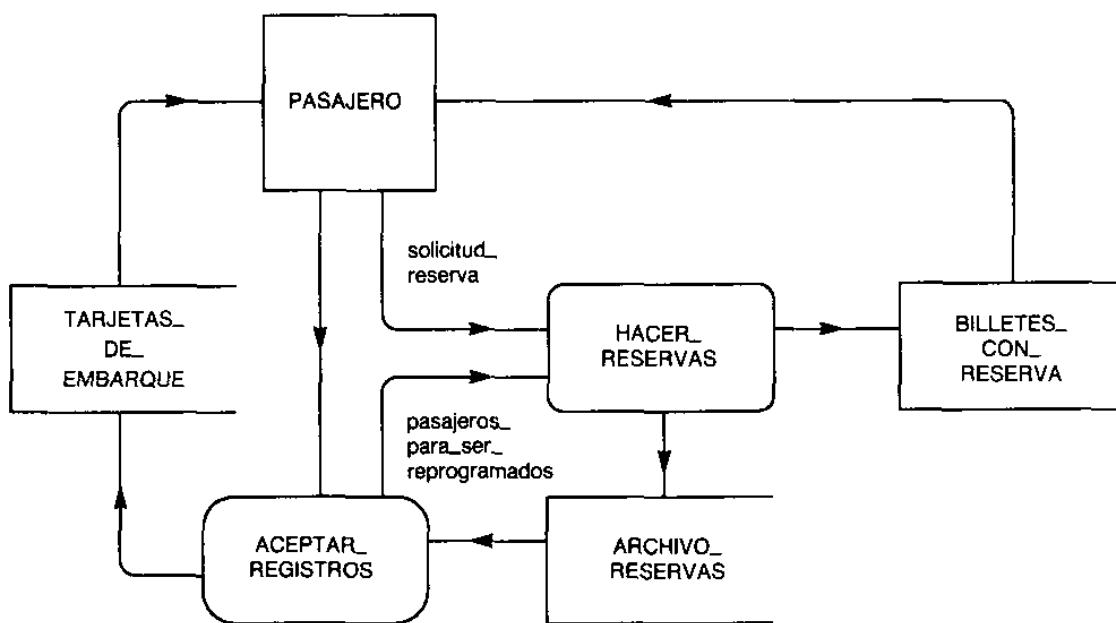
1. Un **proceso** representa una actividad dentro del sistema de información. En

- particular, un proceso puede generar, usar, manipular o destruir información. Cuando un proceso no genera ni destruye información, transforma los datos de los flujos de entrada en datos de los flujos de salida.
2. Un **flujo de datos** es un intercambio de información entre procesos. Se recalca que los flujos de datos *no* representan flujos de control, como la activación de procesos. Indican, en cambio, paquetes discretos de datos que fluyen hacia adentro o hacia afuera del proceso.
 3. Un **almacén de datos** es un depósito de información. Los archivos temporales, tablas de consulta, formularios de papel, formularios electrónicos y registros permanentes, todos éstos pueden representarse como almacenes de datos. Una línea desde un almacén de datos a un proceso indica que los datos del almacén son utilizados por el proceso; una línea desde un proceso a un almacén de datos significa que el proceso modifica el contenido del almacén de alguna manera.
 4. Una **interfaz** es un usuario externo del sistema de información, que puede ser el originador o el receptor de los flujos de datos o de los almacenes de datos.

Para describir un diagrama de flujo de datos, se toma la perspectiva de la organización que usará el sistema de información, por lo que todas las actividades que contribuyen a la manipulación de los datos dentro de la organización se modelan a través de procesos; por otro lado, se hace caso omiso de las actividades que manipulan información fuera de la organización. Las interfaces representan las fronteras del sistema de información, esto es, los generadores o receptores de la información que se manipulan dentro del sistema.

Como primer ejemplo de aplicación de los DFD, se considera el proceso de hacer reservas y preparar los abordajes de vuelos. Para simplificar, se supone que los pasajeros compran billetes cuando obtienen las reservas; cuando se registran, obtienen las tarjetas de embarque, si todavía hay asientos disponibles; sin embargo, debido a la sobreventa de pasajes para los vuelos, es posible que tengan que ser reprogramados para otros vuelos. La figura 8.2a muestra la representación de este ejemplo usando un DFD. Los procesos representan las actividades de HACER_RESERVAS y ACEPTAR_REGISTROS, vistos naturalmente desde la perspectiva de una línea aérea. Así, PASAJERO es una interfaz, mientras que INFORMACION_DE_VUELO, BILLETES_CON_RESERVA y TARJETAS_DE_EMBARQUE representan los almacenes de datos. La actividad de registro termina al darle una tarjeta para embarcar al pasajero o al producirse un flujo de entrada, PASAJEROS_POR_REPROGRAMAR, para la actividad de reserva cuando no hay asientos disponibles.

Obsérvese que los nombres de los conceptos usados en un DFD son esenciales para entender el significado del diagrama. En realidad, toda la semántica es comunicada por los nombres, de modo que éstos deben seleccionarse cuidadosamente. Por ejemplo, se debe evitar nombres como *datos_de_entrada* o *datos_de_salida* para flujos dentro o fuera del proceso, porque esos nombres



(a) DFD para el ejemplo de reservas y registros

HACER_RESERVAS: El pasajero solicita una reserva; si se acepta, se graba la reserva y al pasajero se le entrega un billete.

ACEPTAR_REGISTROS: El pasajero con billete solicita cerrar el vuelo; si todavía existen billetes libres, se le entrega una tarjeta de embarque (*abordaje*); en caso contrario, al pasajero se le asigna un vuelo posterior.

pasajeros_para_ser_programados. El flujo contiene la indicación del pasajero, el destino y el vuelo reservado originalmente.

(b) Glosario para el ejemplo DFD

Figura 8.2. Ejemplo de DFD.

no portan ningún significado. Si se desea añadir semántica más allá de los nombres de los conceptos, es posible desarrollar **glosarios**, esto es, descripciones en lenguaje natural de los conceptos del DFD; los glosarios se introdujeron ya para los conceptos del modelo ER en el capítulo 4. La figura 8.2b muestra el glosario de los procesos **HACER_RESERVAS** y **ACEPTAR_REGISTROS** y del flujo **pasajeros_por_reprogramar**. Los glosarios para los procesos y flujos de datos pueden ser útiles en las fases de seguimiento del diseño; explican el desarrollo de actividades y el intercambio de información dentro del sistema.

El apartado 9.1 describe cómo se puede enriquecer posteriormente un DFD a través de la descripción de los datos que son tratados por los procesos o que requieren los flujos de datos o los almacenes de datos; ésta será la primera etapa en la combinación de los esquemas de datos y de funciones.

8.2. Primitivas para el análisis funcional

Como el análisis conceptual de datos, el análisis funcional tiene el objetivo de producir un esquema global (DFD) de las funciones (procesos) de la empresa. Esta tarea es compleja y generalmente se realiza mediante etapas iterativas. Igual que con el análisis de datos, primero se describen las primitivas para transformar un esquema de entrada en uno de salida, y luego las estrategias progresivas para generar un DFD complejo.

El diseño de un DFD se desarrolla iterando varias **transformaciones de esquemas**, con los siguientes rasgos: 1) cada transformación tiene un *esquema inicial* y un *esquema resultante*, 2) cada primitiva de transformación hace corresponder los *nombres* del esquema inicial con nombres de los conceptos del esquema resultante, y 3) los conceptos del esquema resultante deben heredar todos los *vínculos lógicos* que se definen para los conceptos del esquema inicial. Las transformaciones más simples se denominan **primitivas**; pueden clasificarse como **ascendentes** o **descendentes**. Recuérdese del apartado 3.1 que las primitivas descendentes tienen un solo concepto como esquema inicial, y producen una descripción más rica de ese concepto en el esquema resultante. La figura 8.3 muestra varias primitivas descendentes que se aplican con frecuencia:

1. La primitiva T_1 refina un proceso para dar un par de conceptos conectados por un flujo; se aplica cuando se distinguen en el proceso dos subprocesos, de manera que el primero comunica información al segundo.
2. La primitiva T_2 refina un proceso para dar un par de procesos y un almacén de datos; se aplica cuando los dos procesos pueden ejecutarse en momentos diferentes, así que los datos compartidos deben guardarse en un almacén de datos. Usualmente, el almacén de datos es una porción de la base de datos.
3. La primitiva T_3 divide el proceso en dos procesos independientes; se aplica cuando los dos procesos no están conectados por un intercambio de información inmediato ni diferido.
4. La primitiva T_4 divide un flujo para dar un conjunto de flujos; corresponde a distinguir en el flujo varios tipos de información independientes.
5. La primitiva T_5 refina un flujo para dar dos flujos y un proceso; se aplica cuando se reconocen en el flujo algunas transformaciones escondidas de su contenido de información. Dicha transformación tiene que ser realizada explícitamente por un proceso.
6. La primitiva T_6 divide un almacén de datos en dos almacenes de datos independientes. Se aplica cuando se puede distinguir dos subconjuntos del almacén de datos que están interconectados con diferentes procesos o interfaces.
7. La primitiva T_7 refina un almacén de datos para dar dos almacenes de datos conectados por un proceso; se aplica cuando se puede distinguir dos subconjuntos del almacén de datos tales que uno pueda ser obtenido del otro por medio de un proceso. Esta primitiva es similar a T_5 ; la única diferencia entre ellas es que en T_7 los datos son almacenados permanentemente.

Primitiva	Esquema inicial	Esquema resultante
T ₁ : Descomposición de proceso con flujo intermedio		
T ₂ : Descomposición de proceso con almacén intermedio		
T ₃ : Descomposición de proceso sin conexiones		
T ₄ : Descomposición de flujo		
T ₅ : Refinamiento de flujo		
T ₆ : Descomposición de almacén		
T ₇ : Creación de almacén		

Figura 8.3. Primitivas descendentes para los DFD.

8.3. Estrategias para el análisis funcional

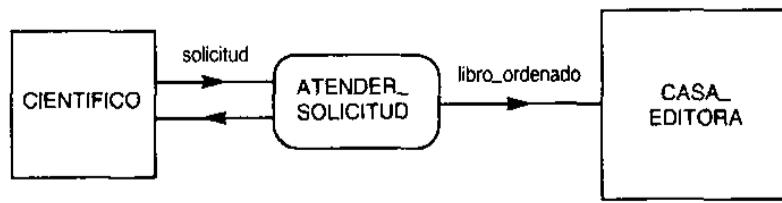
Este apartado presenta las estrategias para el análisis funcional. Como en el caso del análisis de datos, se consideran estrategias descendentes, ascendentes, centrífugas y mixtas. La mayor parte de la literatura sobre análisis y diseño de sistemas (y particularmente los métodos que usan DFD) recomiendan la descendente como la estrategia de diseño más conveniente. Creemos que las otras estrategias también son viables, particularmente las centrífugas y mixtas.

8.3.1. Estrategias descendentes

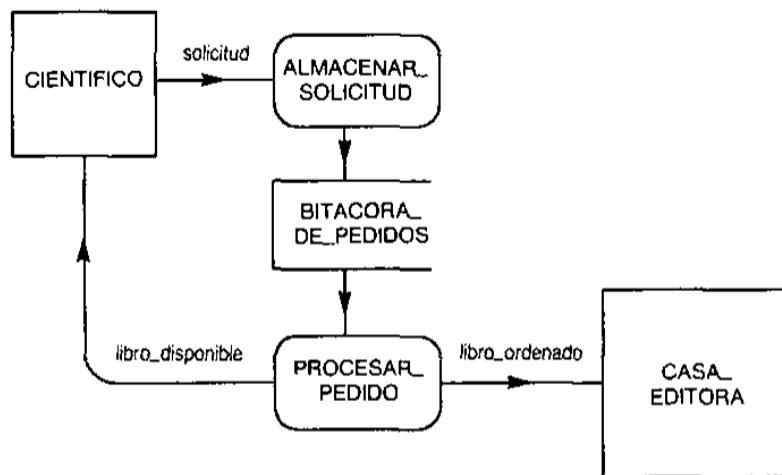
Una estrategia descendente *pura* consiste en la aplicación de las primitivas descendentes puras; cada aplicación produce el refinamiento de un solo concepto. Las primitivas T_1 , T_2 , y T_3 del apartado anterior producen un refinamiento descendente de los procesos y son las más usadas durante el diseño descendente de un DFD. Como el concepto de proceso es el más importante en los DFD, es vital indicar cómo se realiza la descomposición de un proceso en subprocessos. El criterio más importante se denomina **independencia funcional**. La idea clave es descomponer cada proceso en subprocessos bien diferenciados que sean, en lo posible, independientes entre sí. En otras palabras, cada proceso debe identificarse claramente, y las conexiones entre los subprocessos deben ser débiles.

Por ejemplo, considérese el pedido y compra de libros de la biblioteca de un departamento de una universidad. La primera representación de alto nivel de un DFD para este ejemplo, que se muestra en la figura 8.4a, incluye un proceso y dos interfaces. El proceso ATENDER_SOLICITUD modela todas las actividades de pedido y compra. Las dos interfaces son CIENTIFICO, que ordena un libro nuevo y es informado cuando el libro está disponible, y CASA_EDITORA, que recibe los pedidos y vende los libros a la biblioteca. Se procede aplicando refinamientos descendentes al proceso ATENDER_SOLICITUD en etapas sucesivas:

1. Supóngase que la política del bibliotecario es recoger los pedidos de libros y procesarlos periódicamente (por ejemplo, al final de la semana). En este caso una descomposición natural del proceso ATENDER_SOLICITUD se obtiene aplicando la primitiva T_2 y refinándolo en términos de los dos procesos ALMACENAR_SOLICITUD y PROCESAR_PEDIDO. La interfaz entre los dos procesos la establece claramente el almacén de datos BITACORA_DE_PEDIDOS.
2. Supóngase que el procesamiento de pedidos consiste en verificar la factibilidad de la compra, seguido por la compilación de un pedido. Se puede, en consecuencia, aplicar la primitiva T_1 para producir los dos procesos VERIFICAR_LIBRO y ORDENAR_LIBRO (Fig. 8.4c).



(a) Esquema armazón

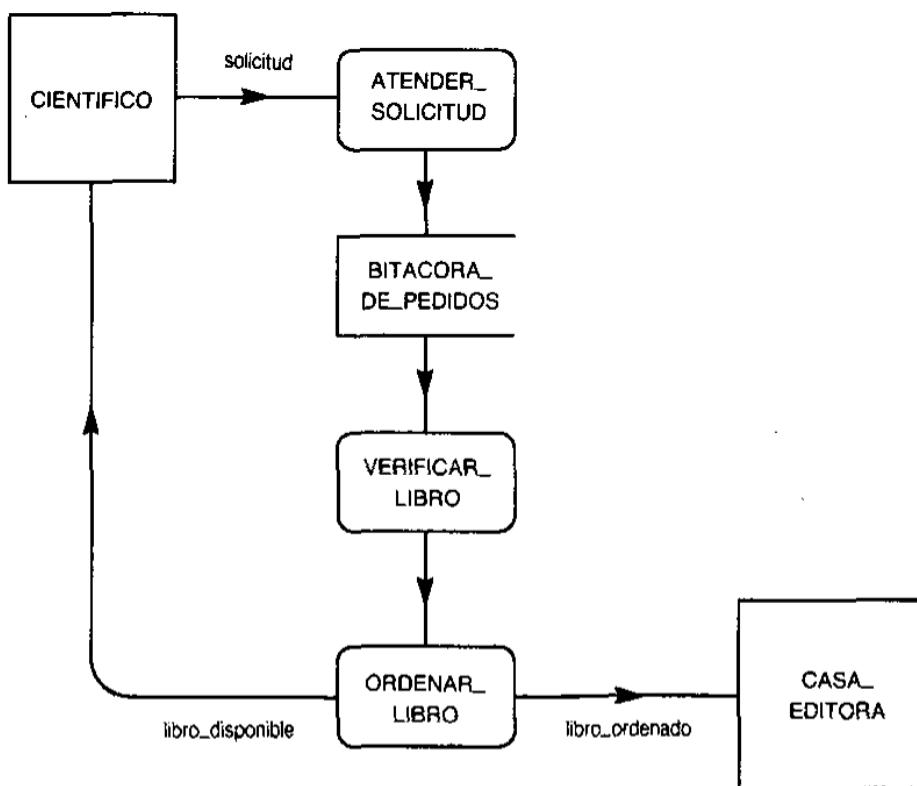
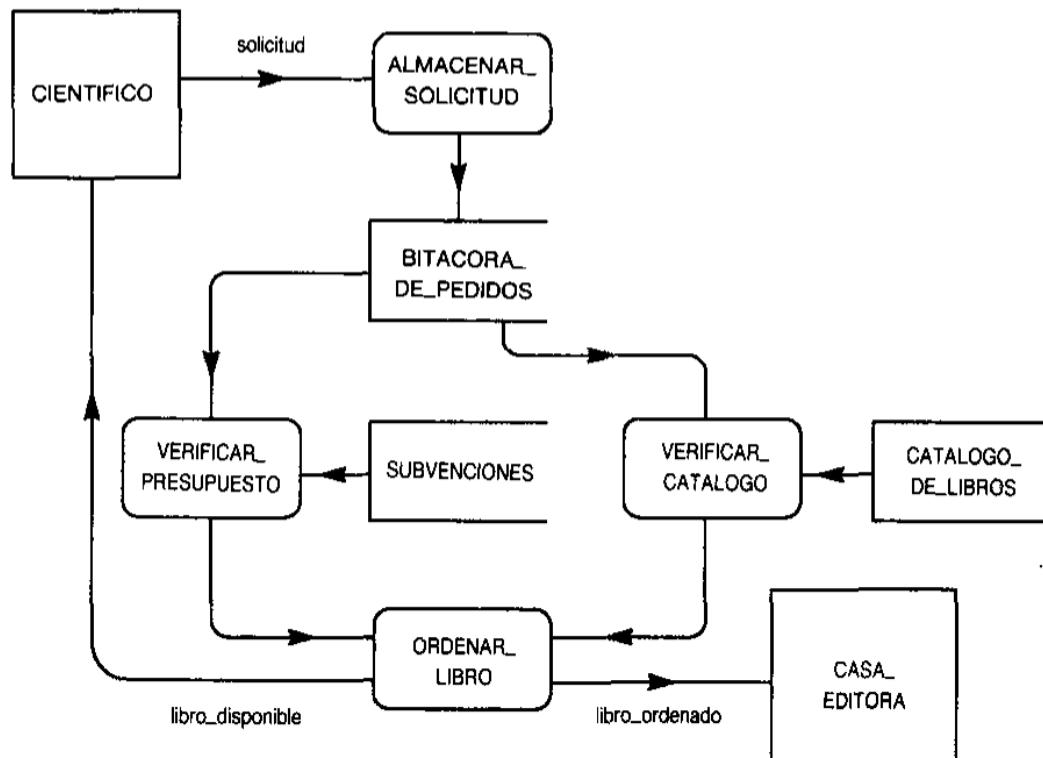
(b) Aplicación de la primitiva T_2 a ATENDER_SOLICITUD**Figura 8.4.** Ejemplo de una estrategia descendente.

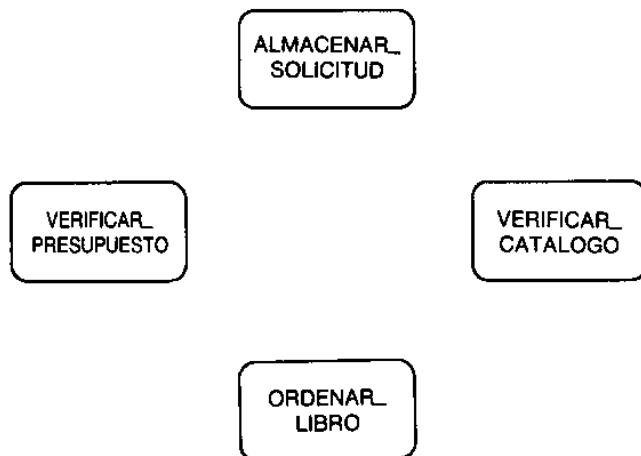
3. Finalmente, supóngase que la actividad **VERIFICAR_LIBRO** puede descomponerse en las dos actividades paralelas de verificar si un libro ya está disponible en la biblioteca y de comprobar si hay fondos disponibles de subvenciones para cubrir el costo. Se puede, por tanto, aplicar la primitiva T_3 al proceso **VERIFICAR_LIBRO**, produciendo los dos procesos **VERIFICAR_PRESUPUESTO** y **VERIFICAR_CATALOGO** (Fig. 8.4d).

8.3.2. Estrategias ascendentes

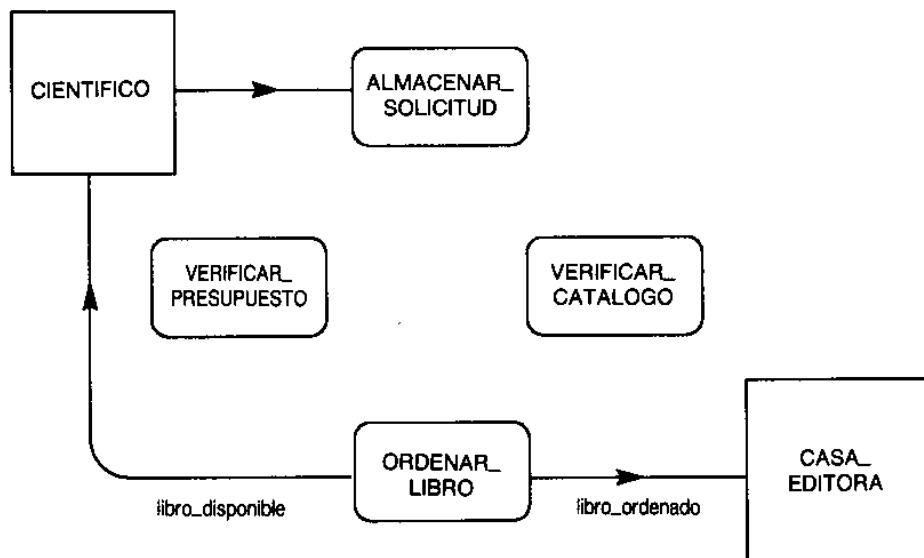
Con una estrategia ascendente, los DFD se construyen a partir de una colección de conceptos elementales, forjando las conexiones entre ellos. El ejemplo previo se desarrolla como sigue:

1. Inicialmente se incluyen en el DFD todos los procesos elementales (Fig. 8.5a).
2. Despues se incluyen todas las interfaces. Tambien se conectan las interfaces a los procesos que intercambian informacion con ellos (Fig. 8.5b).
3. Finalmente, se interconectan los procesos mediante flujos de datos. En par-

(c) Aplicación de la primitiva T₁ a PROCESAR_PEDIDO(d) Aplicación de la primitiva T₃ a VERIFICAR_LIBRO**Figura 8.4.Bis** Ejemplo de una estrategia descendente (continuación).



(a) Procesos elementales iniciales

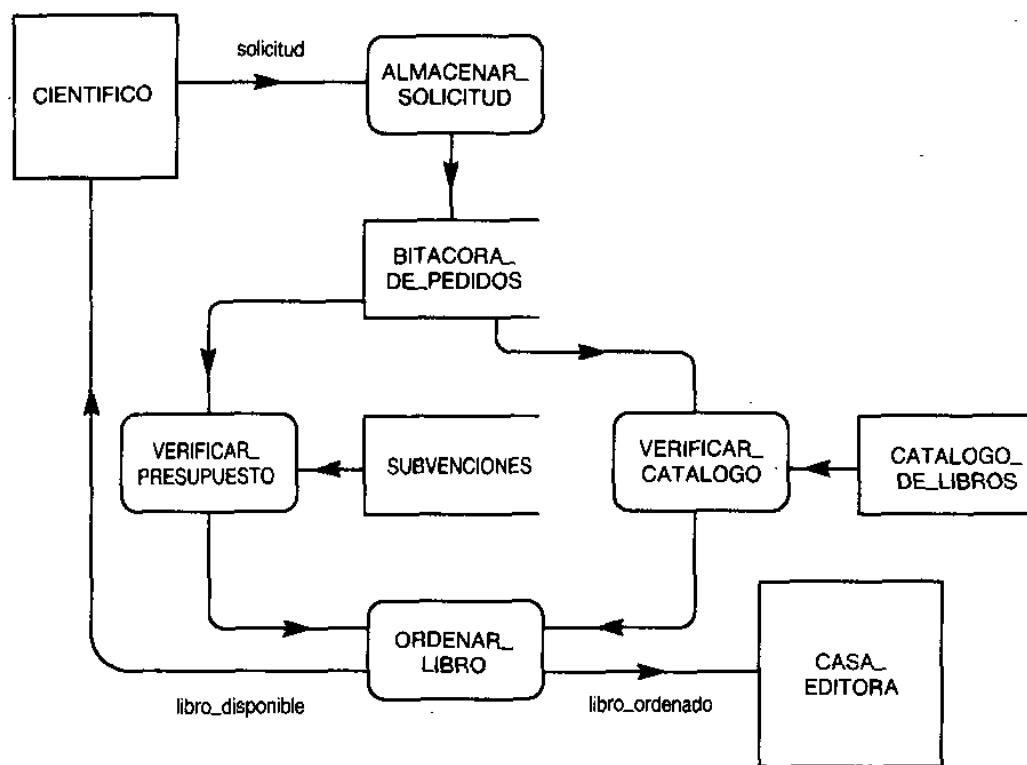


(b) Introducción de interfaces

Figura 8.5. Ejemplo de estrategia ascendente.

ticular, se introduce un almacén de datos cuando la información se registra permanentemente.

Obsérvese que la primera etapa del planteamiento ascendente se define mejor en el análisis de datos, donde se aplica a los atributos, que son los conceptos más elementales del modelo ER. Sin embargo, en el modelo de flujo de datos todos los conceptos están en el mismo nivel; por ello no hay una secuencia específica fija para la construcción de los conceptos en forma ascendente. Se puede, en consecuencia, seleccionar en un orden arbitrario los procesos, interfaces o almacenes de datos.



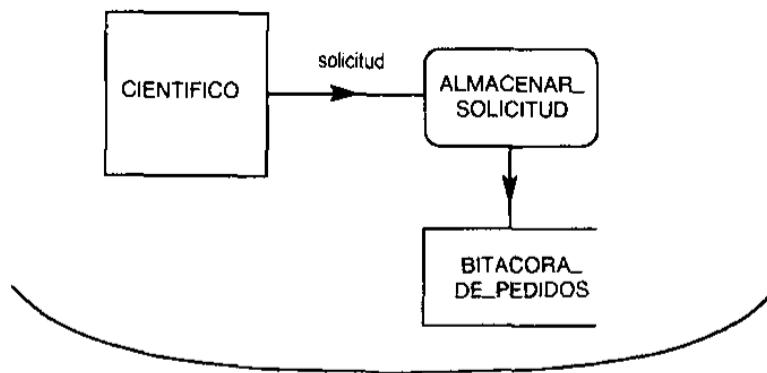
(c) Introducción de flujo de datos y almacenes de datos

Figura 8.5.Bis Ejemplo de estrategia ascendente (*continuación*).

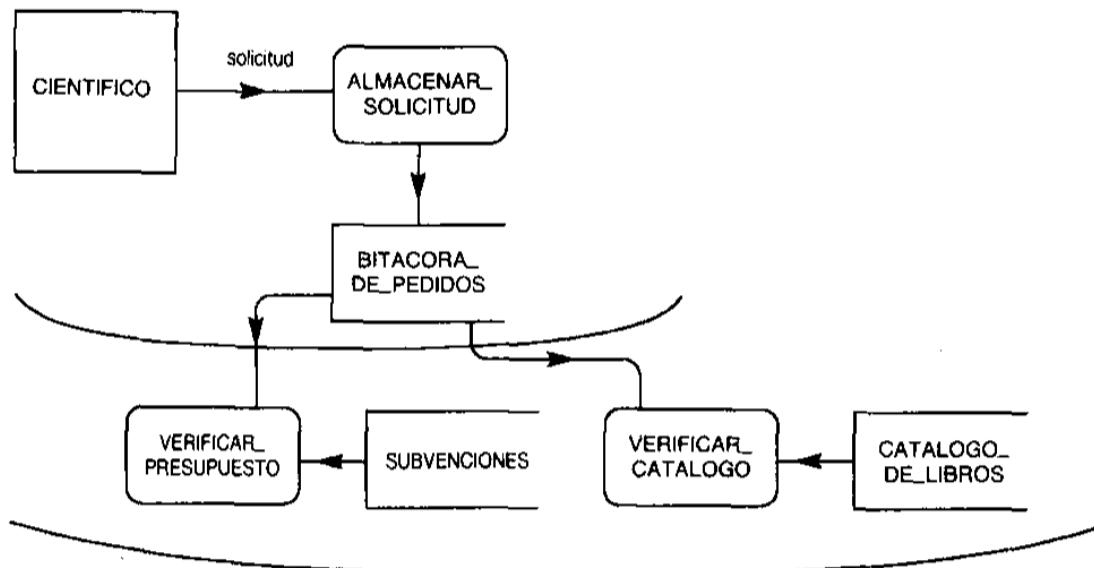
8.3.3. Estrategia centrífuga

La estrategia centrífuga se puede aplicar naturalmente al análisis funcional, porque corresponde a seguir el tratamiento de información que realizan progresivamente los procesos. El término más apropiado para esta estrategia es quizás *centrípeta*, es decir, de afuera hacia adentro y no de adentro hacia afuera, ya que la aplicación más natural de esta estrategia comienza con las interfaces y determina progresivamente los procesos que participan en la producción de los flujos intercambiados con las interfaces. Este proceso se realiza *hacia adelante* si se parte de las interfaces que dan alguna información como entrada al sistema; se realiza *hacia atrás* si se comienza con las interfaces que reciben información que es una salida del sistema. La estrategia hacia atrás también se denomina *orientada a la salida* y es típica de muchas metodologías para el análisis funcional.

La figura 8.6 muestra un ejemplo de estrategia centrípeta hacia adelante. Al principio, se observa que los científicos producen solicitudes que se almacenan en una **BITACORA_DE_PEDIDOS** (Fig. 8.6a). Luego nos damos cuenta de que los pedidos son analizados para ver si hay subvención disponible y para asegurarse de que un libro ordenado no esté ya disponible (Fig. 8.6b). Por último, se ob-



(a) Primera expansión centrífuga



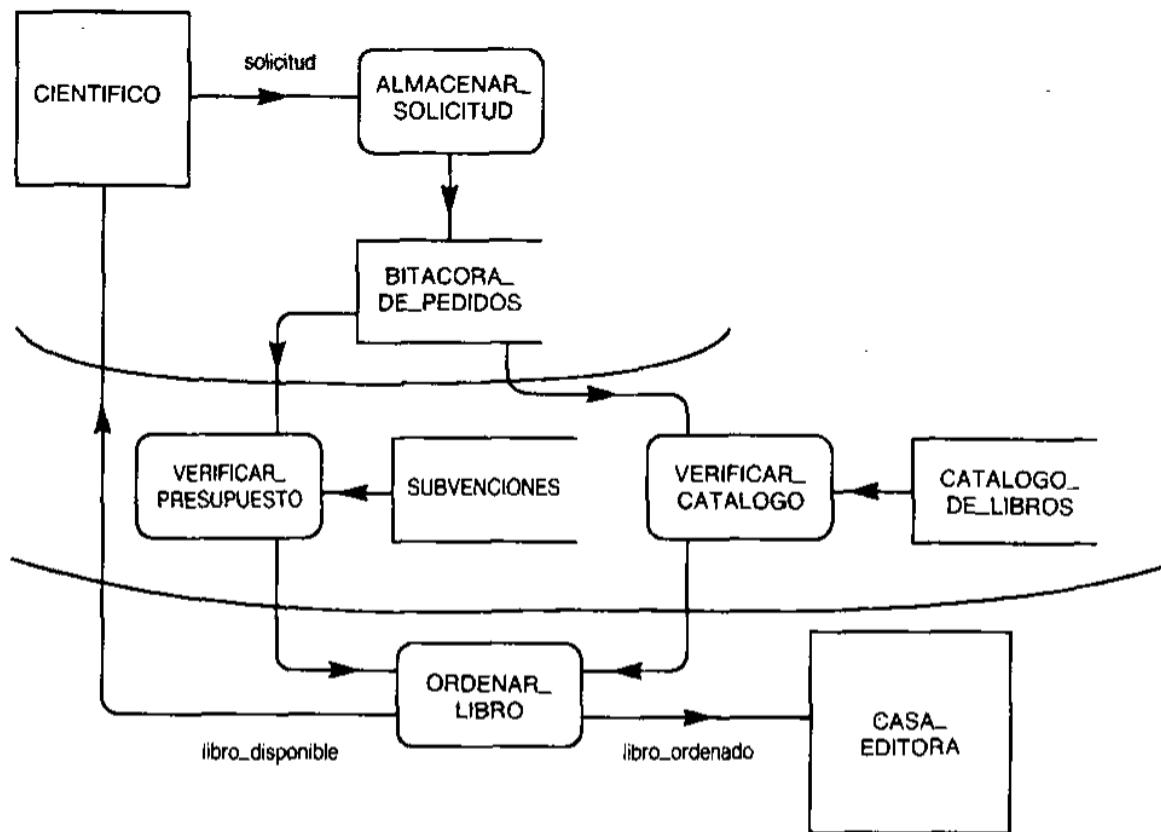
(b) Segunda expansión centrífuga

Figura 8.6. Ejemplo de estrategia centrífuga.

serva que los libros se solicitan a una casa editora si todas las verificaciones son positivas, y luego se les notifica a los científicos (Fig. 8.6c). Adviértase que la propagación de información se marca en la figura con fronteras progresivas que, como en el caso de análisis de datos, indican el avance del esquema inicial al final.

8.3.4. Estrategia mixta

Una estrategia mixta implica producir inicialmente un esquema armazón que incluya los principales procesos y luego realizar un diseño descendente de cada

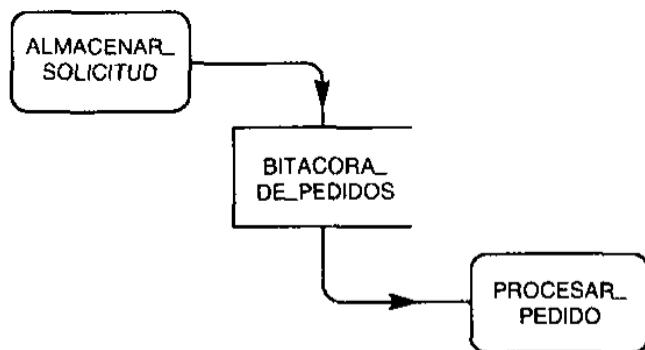


(c) Tercera expansión centrífuga

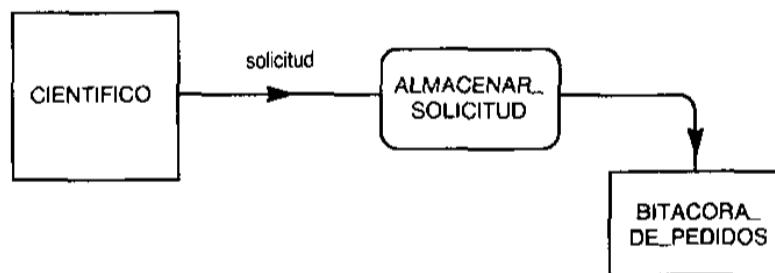
Figura 8.6.Bis Ejemplo de estrategia centrífuga (*continuación*).

uno de esos procesos para producir varios DFD pequeños. Por último, los DFD se integran en un solo DFD; el proceso de integración es dirigido por el esquema armazón. Como en el análisis de datos, una estrategia mixta es conveniente sobre todo si el problema inicial es bastante complejo, porque permite un enfoque de «divide y vencerás».

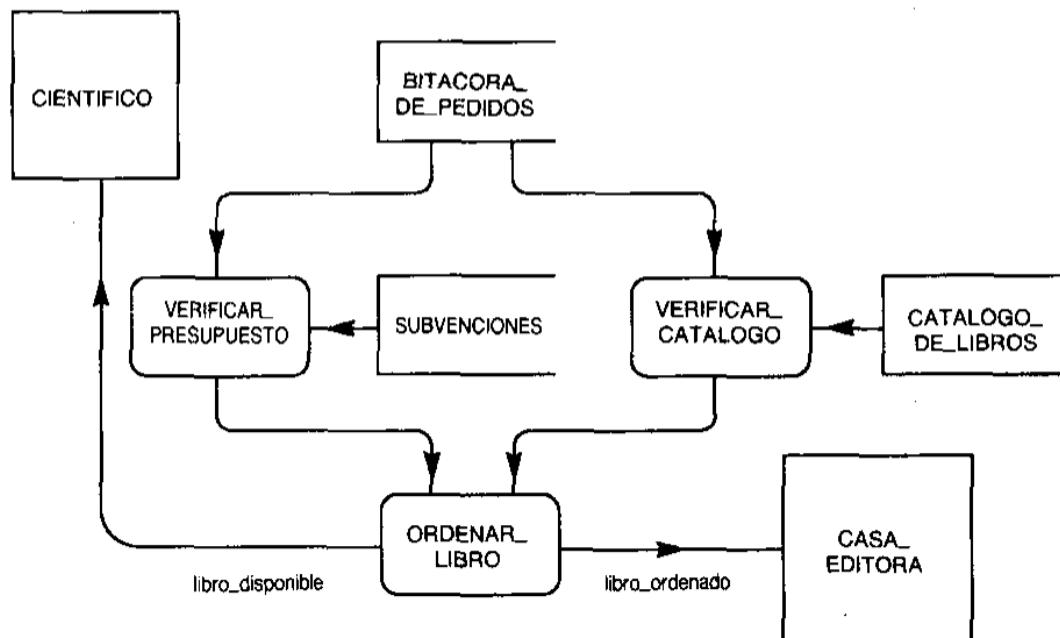
La figura 8.7 muestra un ejemplo de estrategia mixta. El esquema funcional de armazón inicial consiste en sólo dos procesos, ALMACENAR_SOLICITUD y PROCESAR_PEDIDO, que son desacoplados por el almacén de datos BITACORA_DE_PEDIDOS (Fig. 8.7a). El refinamiento del primer proceso da lugar al DFD de la figura 8.7b, que es bastante simple; el refinamiento del segundo proceso da lugar al DFD de la figura 8.7c. Aquí, PROCESAR_PEDIDO se descompone en los tres procesos VERIFICAR_PRESUPUESTO, VERIFICAR_CATALOGO, y ORDENAR_LIBRO. Finalmente, los dos DFD de las figuras 8.7b y 8.7c se fusionan en el DFD de la figura 8.7d, usando el esquema armazón como punto de partida.



(a) Esquema funcional de armazón

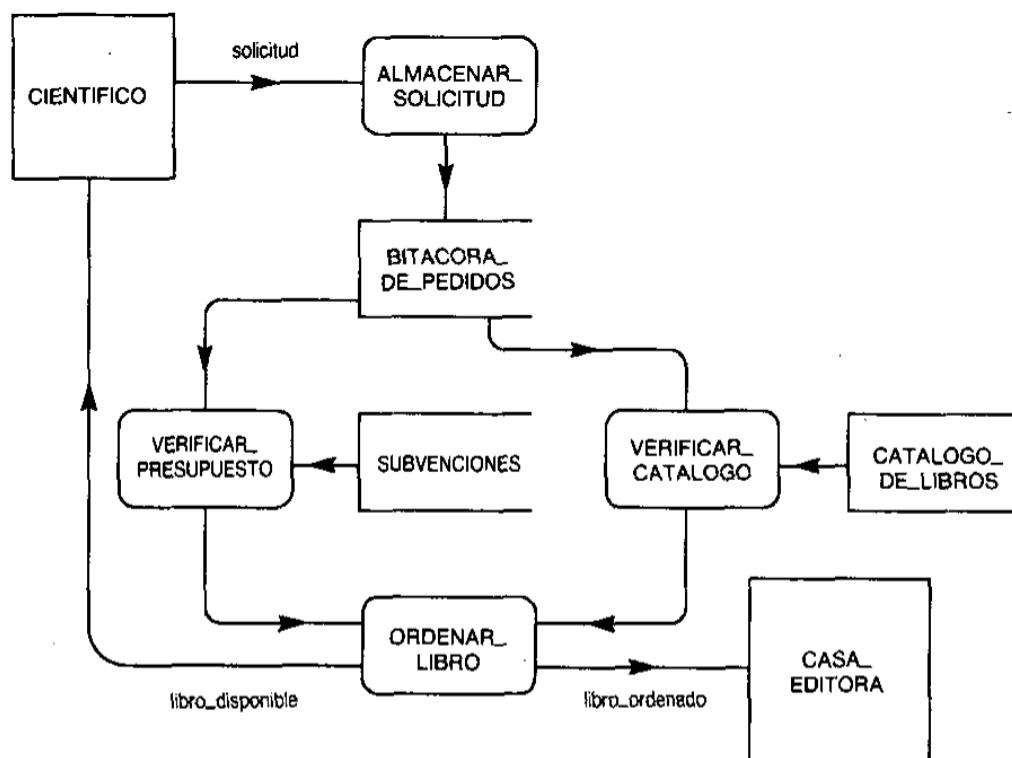


(b) Refinamiento de ALMACENAR_SOLICITUD



(c) Refinamiento de PROCESAR_PEDIDO

Figura 8.7. Ejemplo de estrategia mixta.



(d) Fusión de dos esquemas funcionales dirigida por el esquema funcional de armazón

Figura 8.7.Bis Ejemplo de estrategia mixta (*continuación*).

8.4. Diseño de un esquema funcional

Una metodología para el análisis funcional debe emplear una combinación apropiada de las estrategias expuestas en el apartado anterior. En particular, debe usar la estrategia mixta para el modelado inicial de un DFD armazón, a fin de descomponer el problema. Cada proceso del esquema armazón se debe entonces diseñar usando la estrategia más apropiada; se sugieren las descendentes o las centrífugas como las más convenientes. Tarde o temprano, las cualidades analizadas en el apartado 8.7 deben verificarse en el DFD final. Tal metodología está esbozada en la figura 8.8.

Se considerará brevemente la condición de terminación del análisis funcional. En principio, puede procederse refinando los procesos hasta un nivel muy elemental. Si se consideran los procesos que en última instancia serán ejecutados por un computador, el refinamiento podría iterarse hasta que cada proceso concuerde con un procedimiento o incluso una instrucción elemental. Obviamente, esto no llevará a un buen esquema funcional, por razones metodológicas y prácticas. En primer lugar, desde un punto de vista metodológico, el análisis debe concentrarse en definir *qué* hace un sistema de información, no en *cómo*

1. Identificar interfaces.
2. Identificar flujos de entrada/salida entre interfaces y el sistema.
3. Diseñar un esquema armazón.
4. Repetir

Refinar los procesos del esquema armazón por medio de actividades de diseño descendentes, ascendentes, centrífugas o mixtas,
hasta que
todos los conceptos de los requerimientos se hayan expresado en el esquema, sin describir rasgos procedimentales.
5. Verificar las cualidades del esquema: independencia, compleción, corrección, exactitud, legibilidad y minimalidad.

Figura 8.8. Metodología para el análisis funcional.

opera dicho sistema. Tales consideraciones se dejan para la fase subsecuente del diseño funcional. Por tanto, el esquema funcional resultante del análisis *no debe contener aspectos procedimentales*, que indican cómo se realiza un proceso. En segundo lugar, desde un punto de vista práctico, las redes demasiado complejas de procesos pequeños no son muy útiles porque no producen una visión global manejable del sistema de información. La primera consideración sugiere un criterio para terminar el análisis funcional: no seguir refinando los procesos cuando ello introduce una descripción procedural. Se ilustrará este concepto presentando varios casos en los que los refinamientos de procesos generan rasgos procedimentales.

Considérese el DFD simple de la figura 8.9a, que describe un proceso CONTABILIDAD_DE_PROYECTOS sobre un almacén DATOS_DE_PROYECTOS. Supóngase que se tiene interés en la contabilidad de todos los proyectos dentro de un departamento específico. Luego, un posible refinamiento del proceso CONTABILIDAD_DE_PROYECTOS genera los siguientes procesos: IDENTIFICAR_ELEMENTOS_DE_COSTO_DEL_PROYECTO; REALIZAR_CONTABILIDAD_POR_ELEMENTO, y FUSIONAR_DATOS_DE_CONTABILIDAD. El refinamiento se muestra en la figura 8.9b. Un refinamiento así no es apropiado en el análisis funcional, porque indica el procedimiento que debe usarse para la contabilidad: en particular, el algoritmo que se usa para contabilizar y la navegación dentro de la base de datos requerida para seleccionar la información apropiada.

Como segundo ejemplo, considérese el DFD de la figura 8.10a, el cual describe el proceso de evaluación del peso total de una pieza que tiene subcomponentes. Supóngase que el almacén de datos DESCRIPCION_PIEZA indica los subcomponentes de la pieza o bien, cuando la pieza es elemental, su peso. Así, el peso total se obtiene al sumar los pesos de todos los subcomponentes elementales de una pieza dada. Modelemos esta situación con el proceso EVALUAR_PESO_PIEZA, que opera sobre el almacén de datos DESCRIPCION_PIEZA. Un posible refinamiento de este proceso es generar los nuevos procesos ¿ES_PIEZA_ELEMENTAL? y DETERMINAR_SUBCOMPONENTES_DE_LA_PIEZA (que identifican las piezas elementales), y CALCULAR_PESO_PIEZA_Y_SUMARLO, como se muestra en la figura 8.10b; el primer proceso identifica los subcomponentes y el último lee su peso y calcula la suma del peso.

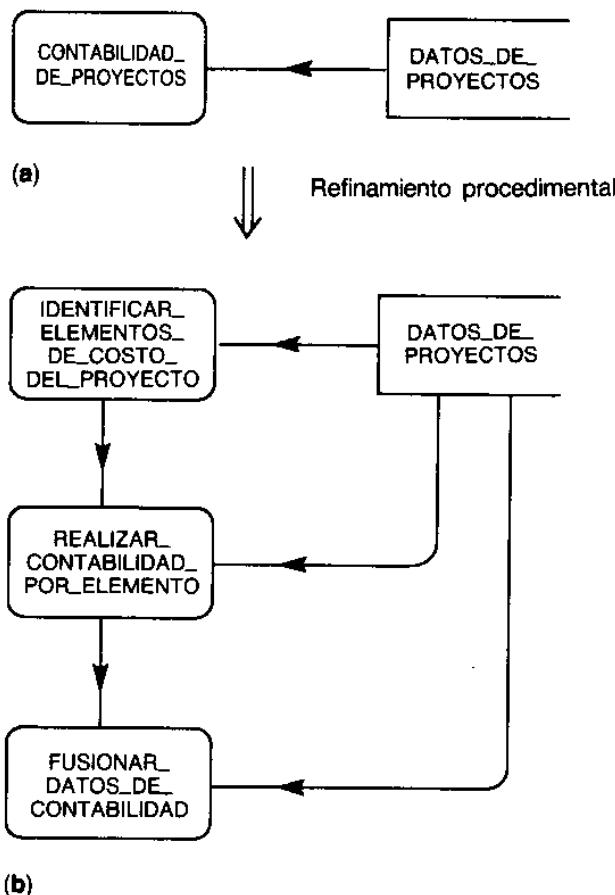


Figura 8.9. Ejemplo de rasgos procedimentales en los DFD: primer caso.

Una vez más, este refinamiento no es apropiado, porque se presentan detalles procedimentales; de hecho, se está modelando los procesos recursivos de descender en la jerarquía de componentes de la pieza hasta las piezas elementales. Obsérvese que en este caso se usa el flujo de datos para modelar recursión. En general, cuando los flujos representan estructuras de control de lenguajes de programación convencionales (por ejemplo, iteración, ciclo, ejecución condicional), es muy probable que se esté describiendo aspectos procedimentales.

8.5. Cualidades de un esquema funcional

Por fin podemos investigar las cualidades de *los buenos DFD*: independencia funcional, compleción, corrección, legibilidad y minimalidad. Todas estas cualidades se aplican también a los esquemas conceptuales (véase Cap. 7) con excepción de la independencia funcional, que es quizás la más importante para el

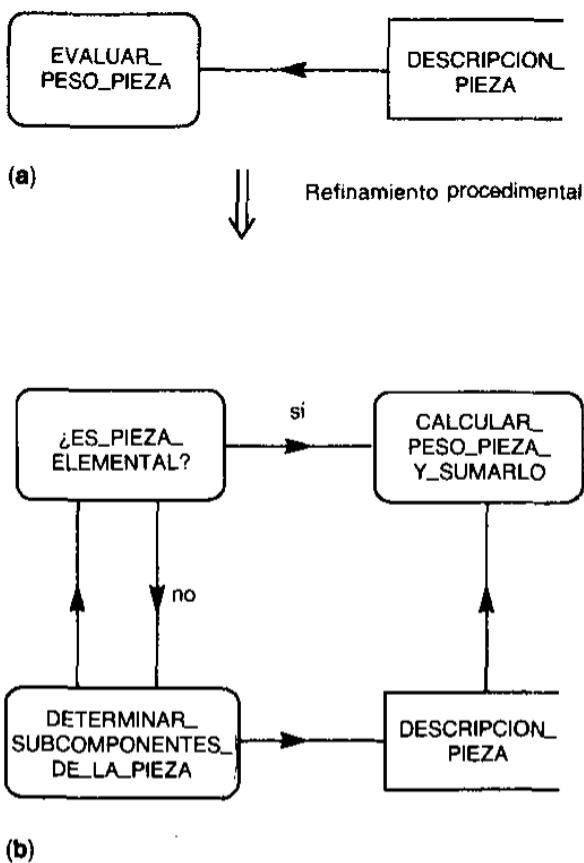


Figura 8.10. Ejemplo de rasgos procedimentales en los DFD: segundo caso.

análisis funcional. Estas cualidades pueden ser difíciles de medir, pero proporcionan guías ideales que deben conducir al diseñador a determinar el DFD más apropiado para representar los requisitos de procesamiento.

Independencia funcional. Esta propiedad se alcanza cuando cada proceso es lo bastante autónomo, es decir, que puede realizar una parte sustancial de sus funciones de manera independiente. Ya se ha subrayado que la independencia funcional debe guiar el refinamiento descendente de los procesos. Cuando se alcanza la independencia funcional, los procesos resultantes tienen las siguientes propiedades adicionales:

1. *Separabilidad:* Cada proceso puede ser analizado en detalle de manera independiente.
2. *Facilidad de integración:* El DFD obtenido como refinamiento de un proceso es fácil de integrar con el resto del sistema.
3. *Flexibilidad:* Cada proceso es adaptable a los cambios sin crear la necesidad de modificar otros procesos.

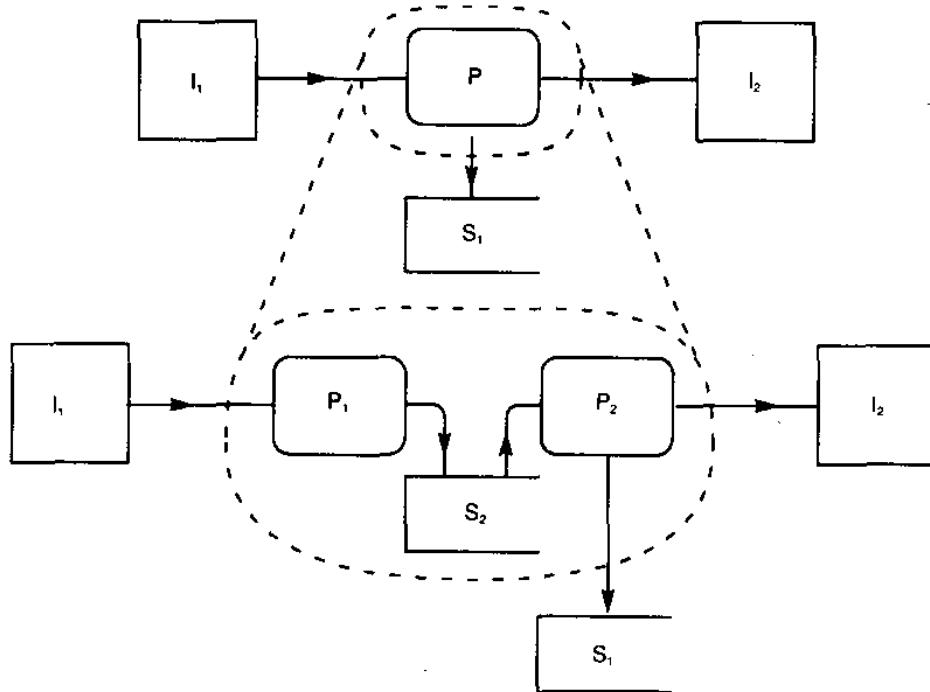


Figura 8.11. Equivalencia en las fronteras después de un refinamiento descendente.

Las propiedades arriba citadas tienen importancia práctica si el diseño se desarrolla por etapas y, en consecuencia, la especificación e implantación de procedimientos que corresponden a los procesos se realizan en momentos diferentes.

Compleción. Un DFD es completo cuando representa todos los rasgos del dominio de aplicación en un nivel de detalle *apropiado* (es decir, cuando se especifican suficientes detalles sobre el mismo pero no se describen rasgos procedimentales). En el próximo apartado se tratará el concepto de compleción mutua entre los datos y los esquemas funcionales.

Corrección. Un DFD es correcto cuando usa de manera apropiada los conceptos del modelo de flujo de datos para representar los requerimientos. También resulta útil indicar cuándo un refinamiento descendente en particular es correcto; para esto, se usa la noción de **equivalencia en las fronteras**. Considérese el refinamiento de un proceso para dar una colección de conceptos y enciérranse estos conceptos en un círculo. La equivalencia en las fronteras se preserva si cada conexión del proceso original con el resto del DFD se hace corresponder a una o más conexiones entre los procesos nuevos que están incluidos en el círculo y el resto del DFD. Tales conexiones se identifican fácilmente porque cruzan la línea fronteriza. Considérese la figura 8.11, en la cual el proceso P experimenta un refinamiento descendente para dar los procesos P₁

y P_2 ; este proceso también produce el nuevo almacén de datos S_2 . Estos conceptos están encerrados en el círculo. La equivalencia en las fronteras se preserva, porque cada línea que entra o sale de P corresponde a una línea que entra o sale de P_1 o P_2 . Si el refinamiento es descendente puro, se espera que ninguna otra conexión cruce la frontera; si, en cambio, el refinamiento no es descendente puro, se aceptarán nuevas conexiones. En metodologías comerciales, la verificación de la equivalencia en las fronteras se denomina **equilibrar el DFD** (véase el apartado 15.4.3).

Legibilidad. Un DFD es legible cuando representa los requisitos de manera natural y puede ser entendido fácilmente sin necesidad de otras explicaciones. Igual que en el caso de los esquemas de datos, se distinguen dos nociones de legibilidad, conceptual y gráfica; la exposición del apartado 6.4 se aplica también a los DFD.

Minimalidad. Un DFD es mínimo cuando cada aspecto de los requerimientos aparece sólo una vez en el esquema. Por ejemplo, cada actividad de la realidad debe corresponder exactamente a un proceso y los almacenes de datos no deben tener partes en común.

Esta exposición de las cualidades esperadas de un DFD nos lleva a una consideración crítica: Es posible, y de hecho probable, que los DFD que reflejan las cualidades arriba citadas no correspondan a la organización del trabajo real en la empresa cuyo sistema de información se está diseñando. Así, la reestructuración de un DFD corresponde a la reestructuración de la organización. De hecho, el diseñador puede utilizar los DFD con dos propósitos distintos: 1) producir una representación de una organización funcional existente (tomando su fotografía instantánea) para superponerle un sistema de información sin alterar las prácticas de organización y operación, o 2) modelar una nueva organización de trabajo que sea más eficiente con respecto a los objetivos de la empresa. No hablaremos intencionalmente de la influencia mutua del análisis funcional y el diseño de organizaciones de trabajo; esta materia está, obviamente, fuera del ámbito de este libro. Pero obsérvese que el material presentado en este apartado da al diseñador los instrumentos técnicos para los dos propósitos.

8.6. Documentación y mantenimiento de un esquema funcional

El planteamiento sobre la documentación y mantenimiento de esquemas que se desarrolló en el capítulo 7 para los esquemas conceptuales se aplica a los esquemas funcionales sin diferencias notables. Los esquemas funcionales deben documentarse de manera descendente. En concreto, el esquema armazón debe incluir un número limitado de procesos complejos, pero debe incluir la mayoría de las interfaces externas (porque éstas no pueden ser producidas por primitivas

descendentes). Los conceptos de DFD del esquema armazón deben entonces expandirse a través de varios planos de refinamiento, manteniendo el equilibrio de esquemas y de conceptos. Realmente, la idea de producir una documentación descendente estricta para los esquemas, explicada en el apartado 7.1, fue desarrollada primero dentro del análisis funcional, donde lo usan muchas metodologías de diseño. Algunas de ellas dan muchos detalles prácticos sobre equilibrio de conceptos; por ejemplo, la metodología SADT, cuyo modelo funcional se repasa en el apéndice de este capítulo, indica que cada proceso de un esquema de refinamiento debe ser expandido a por lo menos tres y, como máximo, seis procesos.

El mantenimiento de programas de aplicación debe entonces realizarse como se muestra en el apartado 7.2, partiendo de la documentación del esquema funcional. Esto permite delimitar la *esfera de influencia* de un cambio de requerimientos, determinar las funciones afectadas y, luego, deducir qué programas de aplicación necesitan ser reespecificados y reimplantados. Tal enfoque conceptual del mantenimiento hace más fácil determinar todas las consecuencias de un cambio en los requerimientos, sin pasar por alto algunas de ellas.

El diccionario de datos puede integrarse también usando un planteamiento similar al descrito en el apartado 7.3. Así, cuando están disponibles múltiples documentaciones de análisis funcional y necesitan integrarse dentro del diccionario de datos, debe determinarse el nivel apropiado de integración de esquemas. Para producir dicha integración y, finalmente, para producir los otros planos de refinamiento, se debe abstraer o bien refinar el plano de refinamiento integrado.

8.7. Resumen

El propósito principal de este capítulo es analizar el análisis funcional dentro del marco metodológico desarrollado en la primera parte de este libro. El lector debe apreciar la considerable dualidad entre el análisis funcional y el de datos, en términos de primitivas de diseño, estrategias, cualidades del esquema, documentación y mantenimiento. Históricamente, los criterios y las metodologías para realizar los diseños se desarrollaban en cualquiera de los mundos y se volvían a utilizar en el otro; por tanto, parece muy natural desarrollar metodologías de diseño duales y también proponer su integración. Ya se han presentado todas las premisas que apoyan un planteamiento conjunto orientado a datos y funciones para el diseño de sistemas de información, mismo que se tratará en el próximo capítulo.

Ejercicios

- 8.1. Teniendo en cuenta la tabla de comparaciones entre las estrategias para el diseño conceptual presentada en el capítulo 3 (Tabla 3.1), presente una tabla de comparación entre estrategias para el diseño de DFD.
- 8.2. Comente la independencia funcional en relación con las propiedades típicas de ingeniería de software, como modularidad y encapsulamiento.
- 8.3. Presente un ejemplo de refinamiento descendente de un DFD que viole la propiedad de equivalencia en las fronteras (véase el apartado 8.5); muestre un ejemplo correspondiente de refinamiento incorrecto en el modelo ER.
- 8.4. Produzca un esquema funcional con por lo menos los siguientes elementos:
 - a) Cinco procesos
 - b) Tres interfaces
 - c) Tres almacenes de datos
 - d) Cinco flujos de datos con nombre

Describa narrativamente la situación representada por semejante esquema funcional. Al final, analice su trabajo e indique qué primitivas y estrategias ha utilizado.

- 8.5. Considere el esquema funcional para el sistema de información de un equipo de fútbol de la figura 8.12. Muestre cómo producir este esquema usando las siguientes estrategias:
 - a) Estrategia descendente
 - b) Estrategia ascendente
 - c) Estrategia centrífuga
 - d) Estrategia mixta
- 8.6. Considere el esquema funcional para el sistema de información de proyectos de investigación presentado en la figura 8.13. Muestre cómo se produce este esquema usando las siguientes estrategias:
 - a) Estrategia descendente
 - b) Estrategia ascendente
 - c) Estrategia centrífuga
 - d) Estrategia mixta

Considere el proceso REALIZAR_EXPERIMENTOS. Descompóngalo en términos de REVISAR_LA_LITERATURA, COMPRAR_EQUIPO, COMPRAR_MATERIALES, EJECUTAR_EL_EXPERIMENTO y OBTENER_DATOS. Muestre las interfaces y almacenes de datos apropiados.

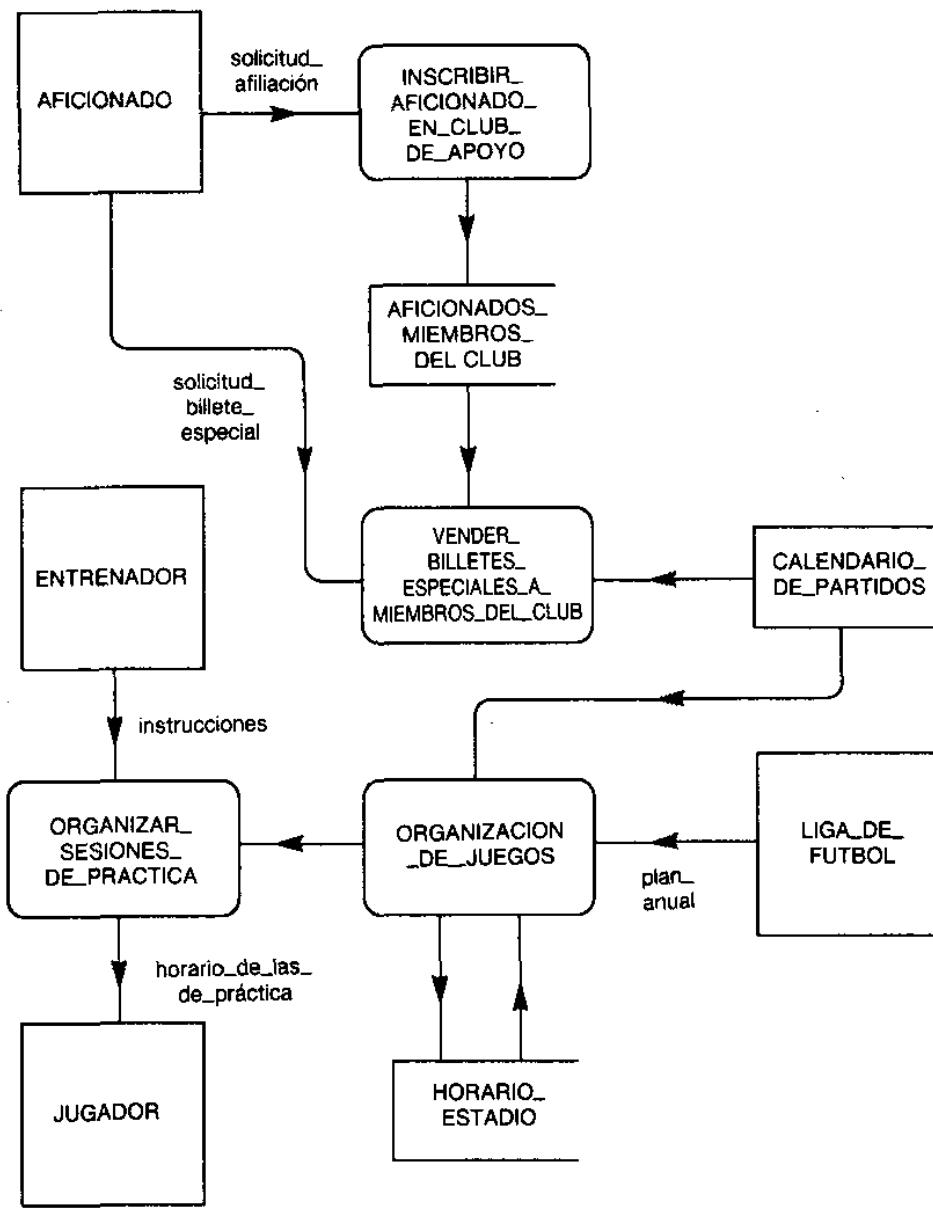


Figura 8.12. DFD de fútbol.

8.7. Para los requerimientos funcionales de una base de datos de un hospital que se presentan en seguida, produzca un esquema funcional usando las siguientes estrategias:

- Estrategia descendente
- Estrategia ascendente
- Estrategia centrífuga
- Estrategia mixta

El sistema de información del hospital tiene un proceso de admisión; los pacientes se admiten en el hospital después de una recomendación de los

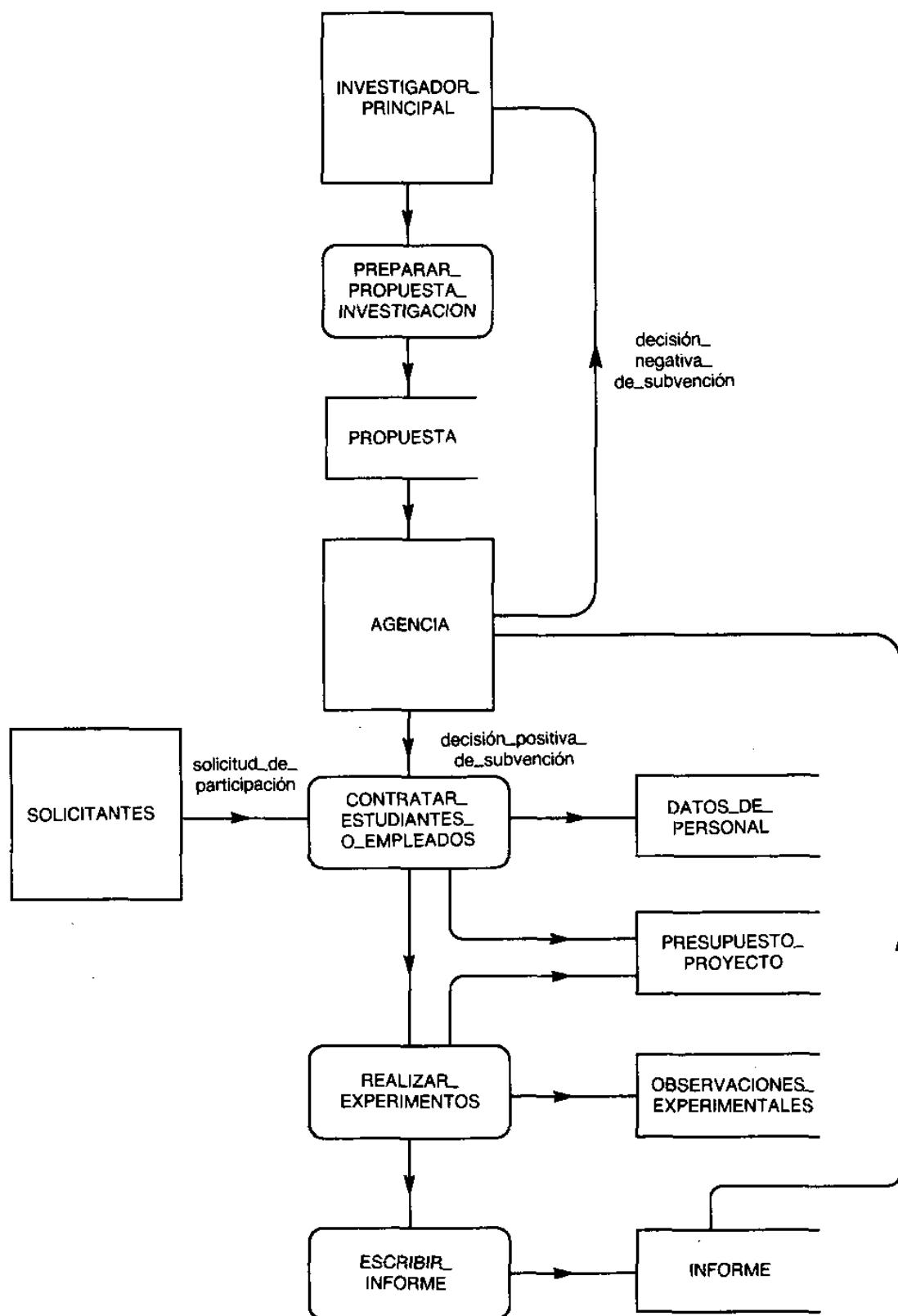


Figura 8.13. DFD de proyectos de investigación.

médicos de admisión. Los datos de admisión de los pacientes se registran; luego, los pacientes se envían al departamento apropiado. A cada departamento se notifica la llegada de pacientes nuevos con una hora de antelación. Los departamentos anotan en la historia clínica del paciente información sobre pruebas, observaciones, tratamientos y reacciones. Antes de darle de alta, los médicos de los departamentos comunican la fecha presunta del alta a los médicos de admisión, de manera que se pueda programar la admisión de nuevos pacientes. En el día del alta, después de un examen cuidadoso, el paciente, en la mayoría de los casos, es dado de alta. A los médicos de admisión se les informa cuando los pacientes tienen que permanecer en el hospital, indicando la nueva fecha presunta de alta.

Apéndice: Repaso de modelos para el análisis funcional

En este apéndice veremos varios modelos que se usan comúnmente en el análisis funcional; todos tienen en común la representación de *procesos*, esto es, transformaciones de datos, y *flujos* de información entre los procesos. Se consideran como elementos primitivos de una vista funcional de un sistema de información. Todos los modelos comentados tienen también una representación gráfica asociada, que es particularmente útil para comunicar los requerimientos al usuario (como con el análisis de datos). Los primeros dos modelos son muy similares a los DFD y suelen usarse para aplicaciones de procesamiento de datos; los últimos dos modelos hacen hincapié en la sincronización de los procesos y se usan generalmente para el modelado de sistemas dependientes del tiempo o sistemas de automatización de oficinas. Cabe señalar que no hay un modelo ideal para el análisis funcional; preferiblemente, el diseñador debe seleccionar el modelo más apropiado dependiendo de los rasgos de su aplicación.

A continuación veremos los modelos en funcionamiento con el mismo ejemplo, a saber, el proceso de HACER_RESERVAS y ACEPTAR_REGISTROS de vuelos que se introdujo en el apartado 8.1. Recuérdese que los pasajeros compran billetes cuando hacen las reservas; al tiempo de registrarse, obtienen una tarjeta de embarque si todavía hay asientos disponibles; pero como un vuelo puede estar sobrevendido, podrían tener que ser reprogramados para vuelos posteriores.

Diagramas SADT

SADT es una metodología para el análisis conjunto de datos y funciones, aunque su enfoque del análisis de datos es muy diferente del que se ha presentado en este libro. El modelo de datos SADT indica sólo cómo participan los datos en los flujos de información entre las actividades, sin tratar la estructura de los datos. El modelo de actividades de SADT se basa en los dos conceptos de actividad y

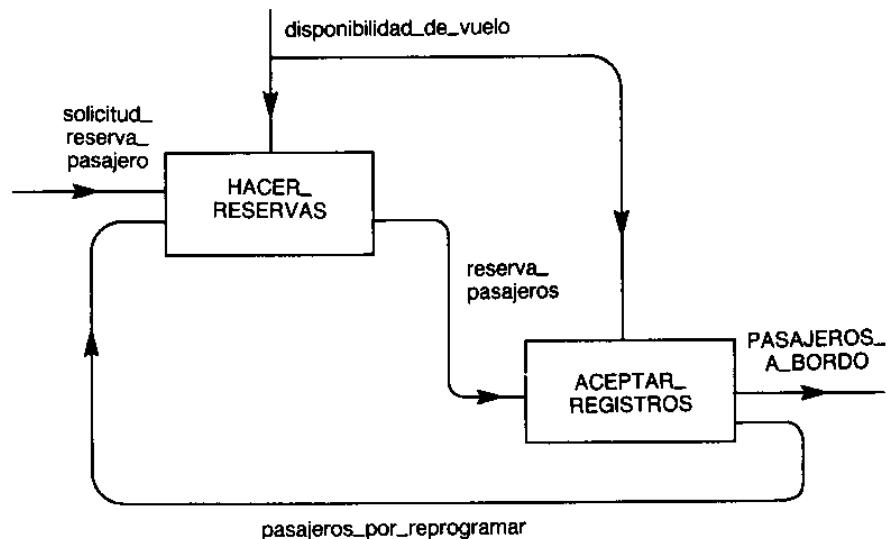


Figura 8.14. Ejemplo de actograma SADT.

flujo de información. Las actividades, que corresponden a los procesos en el modelo de flujo de datos, se representan con rectángulos, y los flujos de información, con flechas. Un actograma SADT debe contener entre tres y seis actividades e incluir todos los flujos de información que las conectan. Los flujos de información se caracterizan por sus papeles: *entrada*, *salida*, *control* y *mecanismo*. Los flujos de entrada corresponden a datos tratados por el proceso; los flujos de salida son datos producidos por el proceso; los flujos de control corresponden a sucesos externos que regulan el proceso; y los flujos de mecanismo, que rara vez se usan, indican información necesaria para el comportamiento normal del proceso.

En un actograma, el papel de un flujo de información respecto a cada actividad se determina por su posición relativa: los flujos de entrada entran por la izquierda, los flujos de salida salen por la derecha, los flujos de control entran por arriba y los flujos de mecanismo entran por abajo. La figura 8.14 muestra un ejemplo de actograma SADT. Comparándolo con la figura 8.2, se observa que las actividades corresponden exactamente a los procesos; también se observa un flujo adicional para el control de actividades, **DISPONIBILIDAD_DE_VUELO**, que no pudo representarse en el DFD.

Gráficas de actividad ISAC (gráficas A)

ISAC es una metodología de éxito para la recolección de requerimientos y el análisis de sistemas de información. En la fase temprana de la metodología, las actividades y funciones se describen a través de gráficas A, similares a los DFD; sin embargo, también representan flujos físicos además de flujos de información.

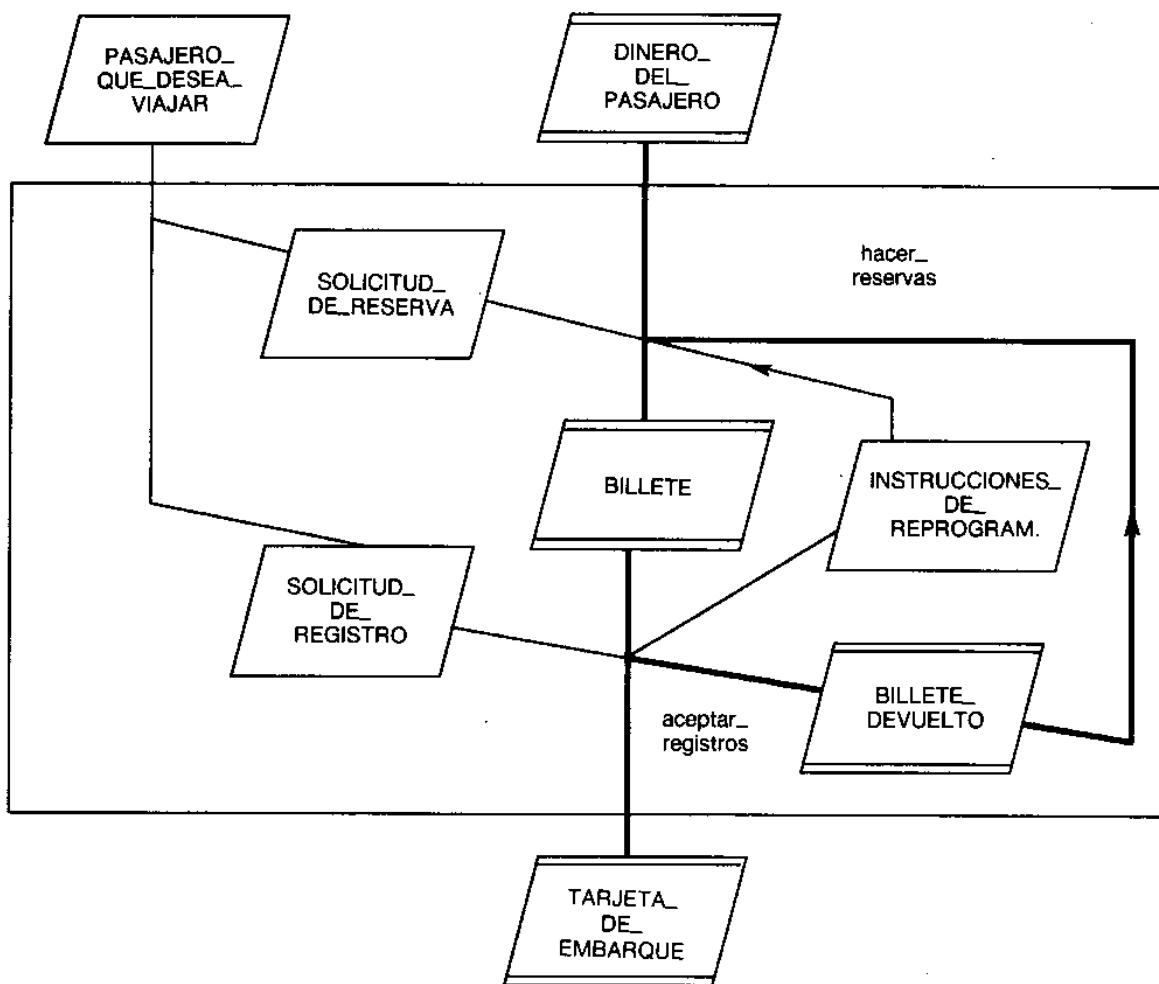


Figura 8.15. Ejemplo de gráficas A ISAC.

Así, los conceptos provistos por el modelo son *conjuntos reales* (materiales o personas), *conjuntos de mensajes* (flujos de información) o *conjuntos mixtos*, junto con los correspondientes flujos reales, de mensajes o mixtos. Los conjuntos reales se representan con rectángulos cuyos lados superior e inferior son líneas dobles, los conjuntos de mensajes, en cambio, tienen líneas sencillas, y los conjuntos mixtos tienen una doble línea arriba y una sencilla abajo. Sólo se dibujan puntas de flecha cuando los flujos van de abajo hacia arriba del diagrama; por defecto, todos los otros flujos van de arriba hacia abajo y no necesitan una flecha. Finalmente, una frontera delimita la porción del sistema de información bajo observación; las entradas están arriba, y las salidas debajo de la frontera. En la figura 8.15, se ve que el proceso total transforma las entradas globales (pasajero inicial con necesidad de viajar y con dinero) en salidas globales (pasajero con tarjeta de embarque, listo para subir al avión). Los flujos reales indican las transformaciones de dinero en billetes de avión y de los billetes en tarjetas de embarque. No aparecen conjuntos mixtos en esta aplicación.

Redes de Petri

Las redes de Petri proporcionan un modelo de flujo de información potente y formal, que resalta los requerimientos de concurrencia de las actividades. Los conceptos provistos por el modelo son los de estados y transiciones. Los estados (o lugares) se representan con círculos, y las transiciones entre los estados, con barras. Las líneas conectan transiciones a estados de entrada y salida. Se interpreta los estados como condiciones y las transiciones como actividades; así, los lugares de entrada (estados) de una transición corresponden a condiciones previas de una actividad, y los lugares de salida (estados) de una transición corresponden a condiciones posteriores. Las fichas indican aquellos estados que están *marcados* o *activos*, y se pueden interpretar como condiciones verdaderas. La dinámica de un sistema de información está representada por el flujo de fichas dentro de la red. Una transición puede dispararse sólo cuando todos sus lugares de entrada tienen al menos una ficha, y su efecto es sustraer una ficha de todos sus lugares de entrada y poner una ficha en todos sus lugares de salida. Las redes de Petri hacen posible una descripción precisa del comportamiento dinámico de un sistema de información; en particular, permite descripciones de actividades en serie, paralelas o conflictivas. El último caso ocurre cuando dos actividades tienen algún lugar de entrada (condición previa) en común y pueden ser disparadas en alternancia. Son posibles varias interpretaciones y variaciones de las redes de Petri; si se desea una descripción más detallada de ellas, véase la bibliografía.

En el ejemplo de la figura 8.16, inicialmente M asientos están disponibles para reservas, es decir, M fichas están en el lugar correspondiente. Cuando un pasajero nuevo entra en escena, se genera una nueva ficha en el lugar correspondiente. También se supone que si un vuelo está listo para salir, se genera una ficha en el lugar correspondiente. Esta ficha tiene el efecto de disparar la transición `abrir_registro`, que a su vez produce N fichas en el lugar `ASIENTOS_DISPONIBLES (N es menor que M si el vuelo está sobrevendido). Cuando a N pasajeros se les da tarjetas de embarque, el vuelo se llena y los otros pasajeros que tienen reservas (cuando más $M-N$) se deben programar para otro vuelo.`

Redes de control de información (ICN)

Las redes de control de información (*information control nets*) han sido desarrolladas para modelar actividades dentro de sistemas de oficinas. Las ICN son similares a las redes de Petri, pero también sustentan el concepto de un depósito de información que es utilizado por una actividad. Las actividades se representan con círculos, las transiciones entre las actividades con flechas de conexión, y los depósitos con rectángulos. Los arcos entre actividades representan relaciones de precedencia: una línea de la actividad A a la B significa que A debe completarse antes de que B pueda empezar. El modelo contiene dos actividades especiales: puntos negros indican puntos de ramificación, y puntos huecos

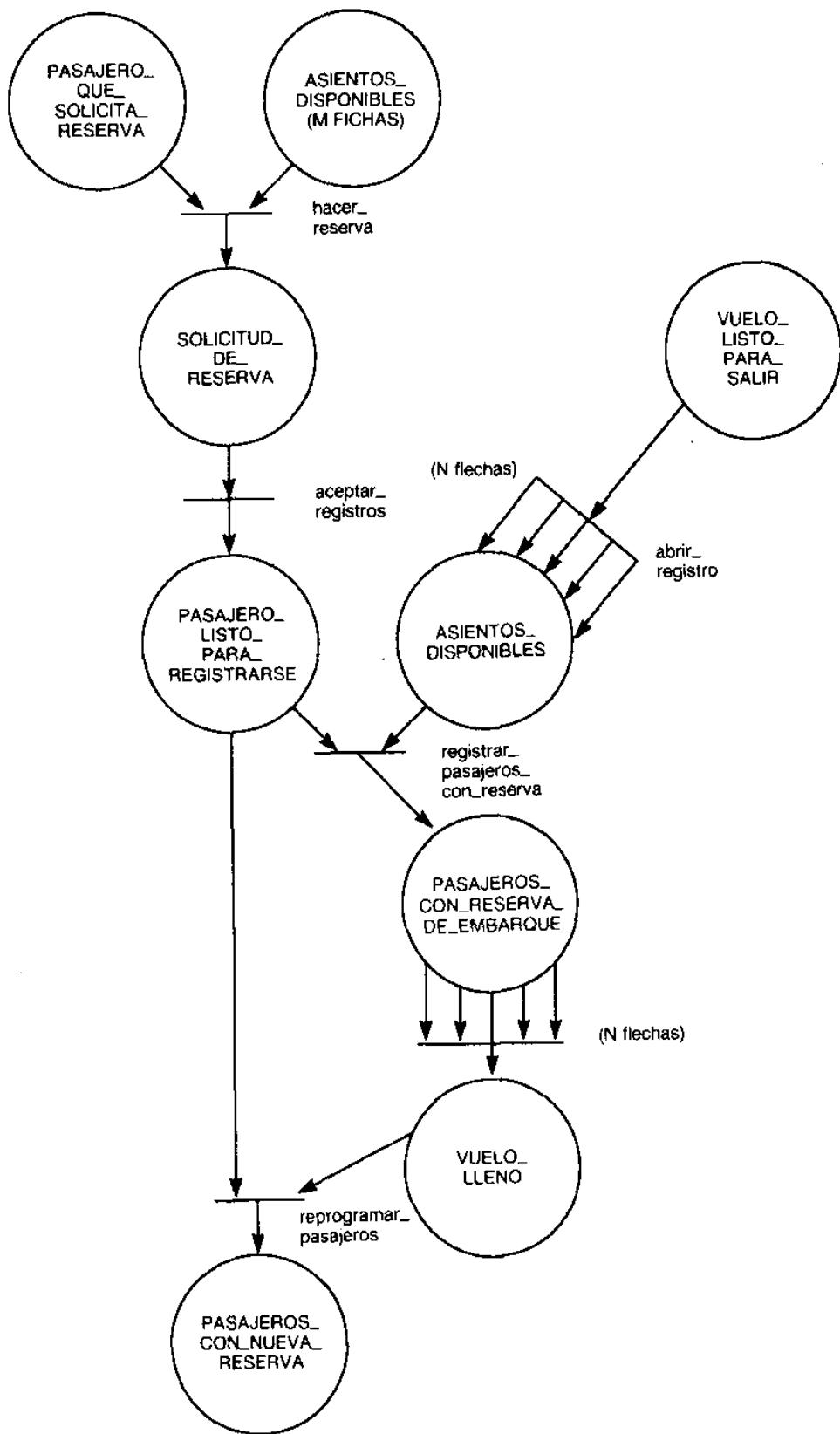


Figura 8.16. Ejemplo de red de Petri.

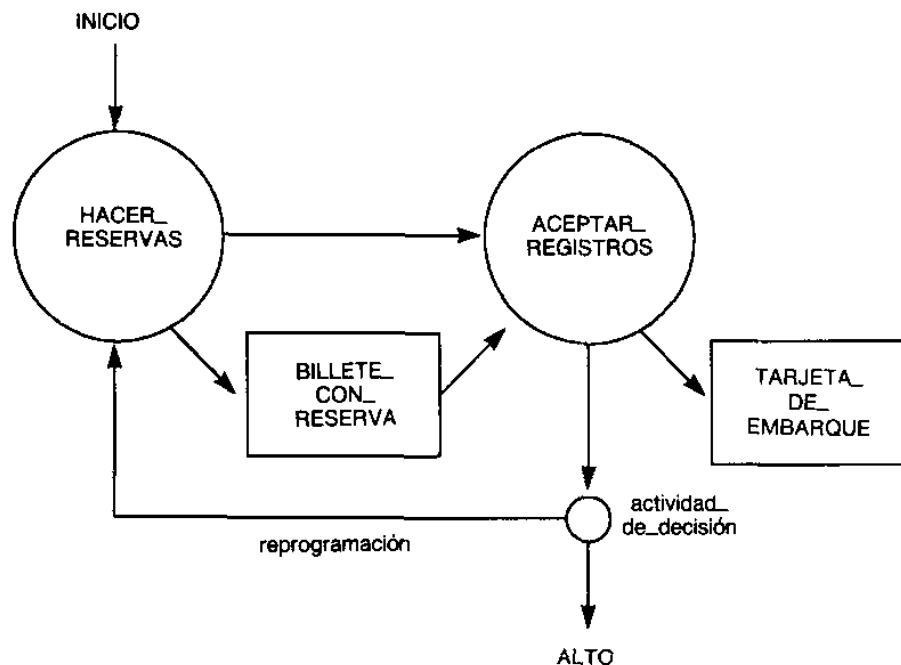


Figura 8.17. Ejemplo de ICN.

representan puntos de decisión. En los dos casos, la ejecución de la actividad es inmediata y sin efecto (actividad «ficticia»). Después de los puntos de ramificación, un número arbitrario de las actividades que siguen puede ser activado ya sea en paralelo o en momentos diferentes; después de los puntos de decisión, sólo una de las actividades que siguen puede ser activada. En nuestro ejemplo, la ejecución ICN puede detenerse (condición ALTO) o puede requerir reprogramación. La figura 8.17 muestra la representación ICN del ejemplo. Obsérvese que la actividad ACEPTAR_REGISTROS va seguida de un punto de decisión, que indica que puede ser completada por dos vías alternativas.

Comparación entre los modelos

La Tabla 8.1 compara los tipos de conceptos representados en los diferentes modelos. Todos los modelos proporcionan conceptos para representar procesos y casi todos suministran conceptos para designar flujos. La representación más simple y más abstracta de estos dos aspectos es la provista por los DFD: ambos, procesos y flujos, se caracterizan simplemente por sus nombres, sin proporcionar información adicional sobre sus papeles o su sincronización. Los defensores de los DFD dicen que esta simplicidad es deliberada, ya que ningún aspecto relacionado con las relaciones causales y de tiempo entre los procesos debe ser expresada durante el análisis: éstos son considerados como aspectos de procedimiento, típicos de la fase de diseño. Los datos en los DFD se representan me-

Tabla 8.1. Comparacion entre los modelos para el analisis funcional

Concepto	DFD	SADT	ISAC	Petri	ICN
Procesos	X	X	X	X	X
Flujo de datos	X	X	X		
Flujo físico			X		
Papel (para flujos)			X		X
Almacén de datos	X				X
Estado				X	X

diente flujos de información o bien mediante información almacenada permanente, destacando así un intercambio dinámico y un depósito estático.

En los diagramas SADT se distinguen cuatro papeles diferentes entre los flujos. Concretamente, los flujos de control permiten expresar hasta cierto grado la procedimentalidad de los procesos; sin embargo, se deja sin especificar la concurrencia de las actividades. Además, los diagramas SADT no mencionan datos almacenados, cuya representación se deja fuera del modelo de actividad. ISAC extiende los tipos de flujos considerados, añadiendo flujos físicos a los flujos de información. Así, las gráficas A de ISAC son particularmente convenientes para la descripción de sistemas de *transformación*, que producen artículos transformando materia prima.

Las redes de Petri puras dan una descripción precisa de concurrencia, porque el modelo se basa en una teoría matemática sólida. Al mismo tiempo, las redes de Petri no incluyen ningún concepto para describir información estática: ni flujos ni almacenes. Por otra parte, las redes de Petri tienden a hacer explícitas todas las condiciones previas y posteriores de las actividades. Esto es ciertamente útil; sin embargo, requiere introducir un gran número de estados y quizás separar una actividad lógicamente única en colecciones de acciones más atómicas, cuando corresponden a condiciones previas y posteriores diferentes. Las redes de Petri se usan convencionalmente para modelar sistemas dependientes del tiempo.

Las ICN se introdujeron como una extensión de las redes de Petri. Los proponentes de las ICN han demostrado que el poder expresivo de las ICN y las redes de Petri para describir la sincronización y la concurrencia es el mismo, mediante correspondencias de un modelo al otro. Sin embargo, el modelo ICN incluye depósitos de información (equivalentes a los almacenes de datos de los DFD). Suelen aplicarse para modelar sistemas de información de oficina, donde se añaden dependencias de tiempo a los requerimientos de procesamiento de datos.

Bibliografía

T. de Marco, *Structured Analysis and System Specification*, Prentice-Hall, 1982.

C. Gane y T. Sarson, *Structured System Analysis*, Prentice-Hall, 1979.

E. Yourdon y L. Constantine, *Structured Design*, Prentice-Hall, 1979.

Estos son los libros más populares en análisis funcional y diseño. Todos usan diagramas de flujo de datos para representar los sistemas de información. El primer libro se centra en la descomposición del proceso de diseño en fases; el segundo y tercer libro dan más detalles sobre el proceso de diseño en sí y también cubren la correspondencia del diseño funcional a las especificaciones de los procedimientos. Nuestra notación de DFD es del libro de Gane y Sarson.

S. Ceri, «Requirement Collection and Analysis in Information Systems Design». En H.J. Kugler, ed., *Proc. IFIP Conference*, North-Holland, 1986.

Este trabajo es la fuente del repaso de modelos para análisis funcional. También indica algunas de las dificultades que caracterizan la aplicación de metodologías a las aplicaciones de la vida real y expone los problemas técnicos y sociales en la recolección de requerimientos y en su análisis.

D. Ross, «Structured Analysis (SA): A Language for Communicating Ideas», *IEEE Transactions on Software Engineering*, SE-3, núm. 1, 1978.

D. Ross y K. Shoman, «Structured Analysis for Requirements Definition». *IEEE Transactions on Software Engineering*, SE-3, núm. 1, 1978.

Softech, Inc. *An introduction to SADT*, Softech, 1978.

Estos trabajos presentan los actigramas SADT en el marco de la metodología SADT para el análisis y el diseño funcionales.

M. Lundberg. «The ISAC Approach to Specification of Information Systems and Its Application to the Organization of an IFIP Working Conference», En T.W. Olle, H.G. Sol, y A.A. Verrijn-Stuart, eds., *Information Systems Design Methodologies: A Comparative Review*, North-Holland, 1982 (Conferencia CRIS 1).

Este trabajo presenta las gráficas A en el marco de la metodología ISAC para análisis y diseño funcionales. SADT e ISAC son quizá las metodologías estructuradas usadas más ampliamente.

C.A. Ellis, «Information Control Nets: A Mathematical Model of Office Information Flow». En *Proc. ACM Conference on Simulation Modeling and Measurement of Computer Systems*, 1979.

J.R. Peterson, «De Petri Nets», *Computing Surveys*, 9, núm. 3, 223-52.

Estos trabajos describen las redes de control de información y las redes de Petri. El segundo da una descripción didáctica y un panorama de los rasgos formales de las redes de Petri, sin centrarse en su aplicación al análisis y diseño de sistemas de información. El primero expone las propiedades formales de ICN y su aplicación al entorno de oficina.

K. Ewusi-Mensah, «Identifying Subsystems in Information Systems Analysis», *Information Systems*, 9, núm. 2, 1984, 181-90.

Este trabajo expone los criterios para descomponer sistemas en subsistemas; algunos de ellos han sido resumidos en nuestra exposición sobre la independencia funcional de los procesos.

- A. Borgida, S. Greenspan y J. Mylopoulos, «A Requirement Modeling Language and Its Logic». En *Proc. Fourth Scandinavian Conference on Conceptual Modeling*, Ellivuori, 1985.

Este trabajo proporciona un lenguaje formal de alto nivel para la especificación de datos y funciones en el nivel de análisis. Los rasgos del lenguaje incluyen clases de aser-ciones, el tratamiento de tiempo y varias técnicas de abreviación, todo integrado en un marco uniforme orientado a objetos.

- O. Barros. *Modeling of Information Systems*, Informe interno núm. 86/07/c. Universidad de Chile, Santiago, 1986.

Este trabajo presenta un enfoque interesante para modelar sistemas de información que tiene en cuenta componentes organizativos externos al sistema de cómputo; el tra-bajo usa un modelo generalizado y técnicas de teoría de sistemas.

Análisis conjunto de datos y funciones

En este capítulo se enfoca el diseño conceptual y el análisis funcional de manera conjunta. Se extienden los DFD para incorporar los **esquemas externos**, a saber, la descripción de ER de los datos que se usan en los procesos o que se guardan en los almacenes de datos. Después se introduce una metodología conjunta para el diseño de datos y funciones, en la que los desarrollos de los esquemas de datos y de funciones influyen uno sobre el otro a través de refinamientos mutuos y verificaciones de compleción. Finalmente se muestra el uso de los esquemas externos como punto de partida para especificar operaciones de bases de datos, que son esenciales para seguir el diseño lógico y físico.

Desde un punto de vista terminológico, la expresión *análisis de datos* se usa para resumir todas las actividades que previamente se han denominado diseño *conceptual* de base de datos, porque este término es normalmente usado en contraposición con el de *análisis funcional*. Se usan los términos **esquema D** para designar un esquema conceptual (de base de datos) y **esquema F** para denotar un esquema de funciones. La idea básica de la metodología de diseño conjunta es alternar los refinamientos del esquema D y del esquema F; cada refinamiento es influído por la versión previa de ambos esquemas, el D y el F. En concreto, se ilustran algunos refinamientos propuestos después de aplicar transformaciones primitivas a esquemas elementales. Después de cada par de etapas de refinamiento realizadas sobre el esquema D y el F, una verificación de congruencia mutua asegura que las dos especificaciones son coherentes. La prueba de la congruencia mutua se facilita por el uso de esquemas externos, que describen la estructura de los almacenes de datos y los requerimientos de datos de los procesos.

En seguida, los procesos se analizan para determinar las operaciones de base de datos, que son pequeñas unidades de interacción con la base de datos. En la práctica, cada proceso puede incluir varias operaciones; por otro lado, una operación de base datos puede ser usada por varios procesos. Las operaciones se especifican usando **esquemas de navegación**: esquemas ER anotados con símbolos especiales que muestran cómo se obtiene acceso a las entidades y cómo se

recorren las relaciones. Los esquemas externos se usan como punto de partida para la extracción de los esquemas de navegación.

El apartado 9.1 presenta los esquemas externos; el apartado 9.2 introduce el enfoque conjunto del análisis de datos y funcional, y aplica este enfoque a un pequeño caso de estudio. El apartado 9.3 sugiere cómo escoger los refinamientos para los esquemas D y F y cómo retener el beneficio de los refinamientos previos en el otro esquema. El apartado 9.4 explica cómo seleccionar las operaciones de base de datos a partir de los procesos y cómo asociarlas con los esquemas de navegación. Los diversos métodos que se presentan en este capítulo se aplican en el siguiente a un caso de estudio más grande.

9.1. Esquemas externos para los diagramas de flujo de datos

En la literatura de bases de datos, el término *esquema externo* se usa para designar una vista particular de la base de datos que se presenta a una aplicación específica o a un grupo de aplicaciones. En una arquitectura compleja de base de datos, varios esquemas externos se pueden construir sobre un esquema conceptual único; cada esquema externo incluye sólo los datos concernientes a la aplicación o aplicaciones consideradas. Así, en la terminología ER, un esquema externo puede omitir porciones enteras del esquema conceptual e incluir solamente aquellas entidades e interrelaciones que se usan en la aplicación pertinente. El esquema externo puede también omitir de las entidades e interrelaciones aquellos atributos que no son mencionados por la aplicación. En general, los esquemas externos se definen para bases de datos en operación; no se usan durante el diseño conceptual de bases de datos.

En esta sección se propone el uso de esquemas externos durante el diseño funcional. En este marco, un esquema externo es un esquema ER que incluye sólo los datos que conciernen a una porción específica de un DFD; en concreto, se propone asociar un esquema externo con algunos de los procesos y almacenes de datos de un DFD. Por ello, algunas veces nos referiremos a él como **esquema externo de proceso**. Diseñar un esquema externo es, por tanto, un caso muy simple del diseño conceptual, cuyo objetivo es desarrollar un esquema ER pequeño, pero completo y correcto.

En la figura 9.1 se presentan algunos ejemplos de esquemas externos para el DFD que se utilizó en el último capítulo, con objeto de mostrar las diversas estrategias de diseño de DFD. La figura 9.1a muestra el esquema externo asociado con el almacén de datos BITÁCORA_DE_PEDIDOS, un archivo que guarda solicitudes de libros hechas por científicos; como tal, el esquema externo correspondiente (usando el modelo ER) tiene las dos entidades LIBRO y CIENTIFICOS, conectadas por la interrelación SOLICITADO_POR. De manera similar, las figuras 9.1b y 9.1c muestran los esquemas externos de los almacenes de datos SUBVENCIONES y CATÁLOGO_DE_LIBROS y la figura 9.1d muestra el esquema externo del

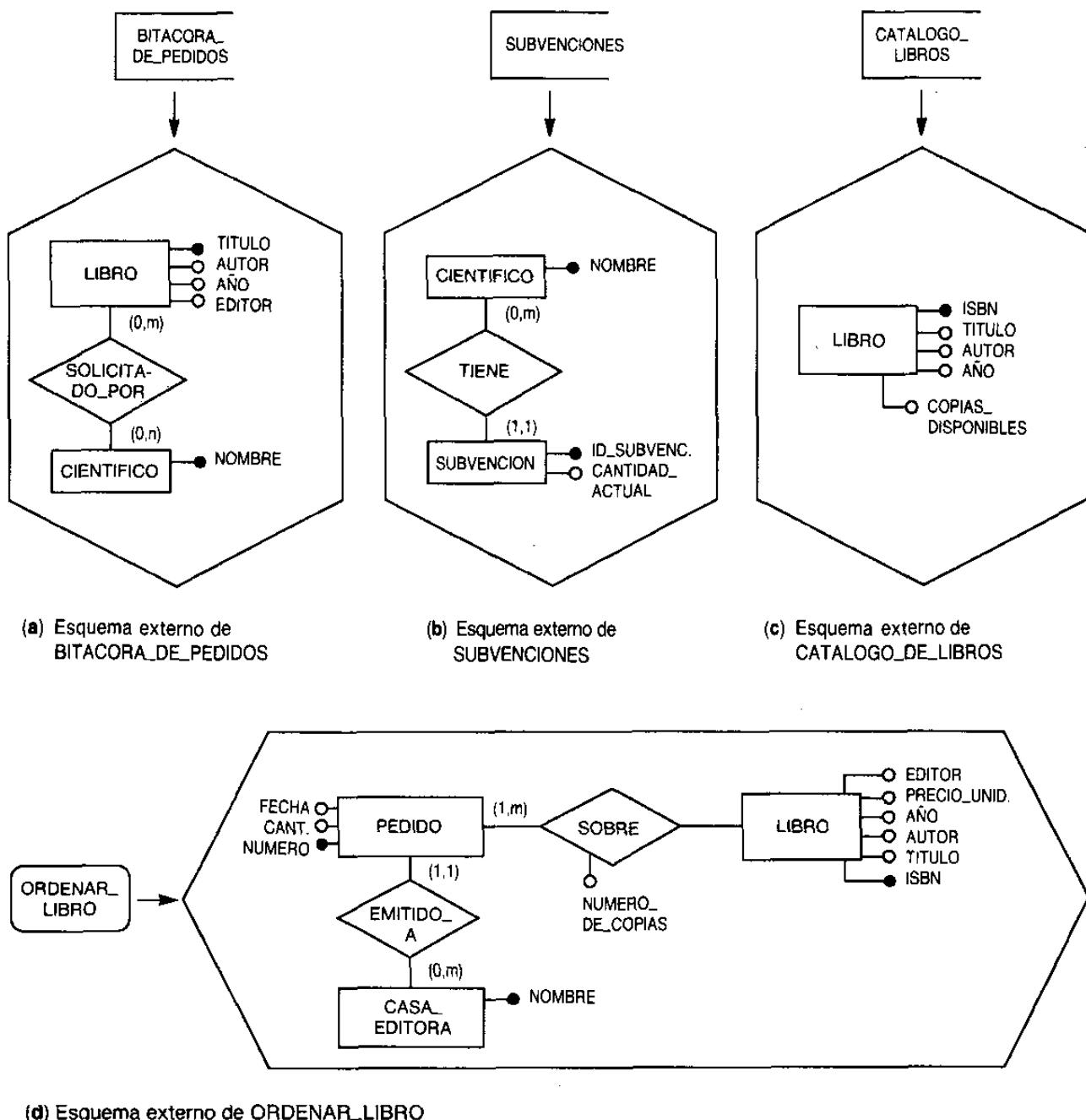


Figura 9.1. Esquema externo para el DFD de la figura 8.4d (y también 8.5c, 8.6c, 8.7d).

proceso ORDENAR_LIBRO. Obsérvese que cada esquema externo está enmarcado en un hexágono conectado con uno de los elementos del DFD.

La introducción de esquemas externos durante el análisis funcional tiene varias ventajas:

1. Ayuda a identificar los datos que deben llegar a formar parte de la base de datos; tal conocimiento puede influir sobre los refinamientos del esquema D.
2. Los esquemas externos sirven para realizar cuidadosas verificaciones de compleción del esquema D, pues aseguran que todos los conceptos de los esquemas externos estén también mencionados en el esquema D.
3. Los esquemas externos pueden llegar a ser útiles para una fase subsecuente del diseño: la identificación y especificación de las operaciones de base de datos.

Si adoptamos una posición extrema, podemos imaginar una metodología ascendente pura de diseño de datos, que se obtiene al diseñar primero cada esquema externo y, luego, integrarlos todos en el esquema conceptual, siempre y cuando cada parte del esquema sea parte de un almacén de datos o lo use un proceso. Sin embargo, una metodología así tiene todos los inconvenientes del diseño ascendente, incluidas la falta de una visión global de los requerimientos de datos y la necesidad de una resolución frecuente de conflictos, además de indeseables efectos colaterales. Se propone, en su lugar, que los esquemas externos sean usados para la verificación de compleción del esquema conceptual, como se mostrará en el próximo apartado.

9.2. Una metodología para el análisis conjunto de datos y funcional

En el capítulo 8 se observaron varias similitudes entre el análisis funcional y el diseño conceptual de base de datos; esas similitudes sugieren que el análisis de datos y funciones puede realizarse de manera conjunta. La principal razón para un análisis conjunto es que cada planteamiento puede apoyar y completar al otro. El proceso de analizar un sistema de información se realiza a través de refinamientos progresivos, y éstos se simplifican si las especificaciones de los datos y funciones se producen conjuntamente. Así, tenemos que los refinamientos de los esquemas de datos pueden sugerir refinamientos de los esquemas de funciones, los que a su vez pueden sugerir refinamientos de los esquemas de datos; este proceso puede repetirse varias veces. Esto es similar a lo que hacen los escaladores de montañas, que normalmente alternan el movimiento de los miembros izquierdos y derechos; cada movimiento hace posible el siguiente.

Una segunda razón para adoptar un enfoque integrado es que ayuda a alcanzar la **compleción mutua**. Se puede estar seguro de que los esquemas de datos y funciones son mutuamente completos sólo si se hace una comparación cruzada de su contenido de información. Por ejemplo, se verifica que los almacenes de datos de los DFD estén representados en el esquema conceptual, e igualmente que las operaciones de manipulación de datos pertenezcan a algún

1. Identificar las interfaces.
2. Identificar los flujos de entrada/salida entre las interfaces y el sistema.
3. Identificar un primer esquema de datos armazón y un esquema funcional armazón.
4. Repetir
 - 4.1. Refinar el esquema D a partir del esquema D previo, observando simultáneamente el esquema F anterior.
 - 4.2. Refinar el esquema F a partir del esquema F previo, observando simultáneamente el esquema D anterior.
 - 4.3. (Paso opcional) Probar la congruencia mutua de los esquemas D y F,
- hasta que
todos los conceptos de los requerimientos hayan sido expresados en el esquema.
5. Verificar las cualidades de los esquemas D y F: compleción propia y mutua, corrección, legibilidad, minimalidad, independencia funcional (sólo para el esquema F); modificar los esquemas D y F para que tengan las cualidades arriba citadas.

Figura 9.2. Metodología para el análisis conjunto de datos y funcional.

proceso. La compleción mutua es una propiedad requerida solamente en las versiones finales del esquema; aun así, se puede alcanzar de manera más fácil (y puede realmente guiar el proceso de refinamiento) si es probada continuamente en las versiones intermedias de los esquemas.

Una metodología descendente de análisis conjunto de datos y funciones se muestra en la figura 9.2. En este enfoque el esquema de datos y el de funciones (abreviados a esquema D y esquema F) se diseñan concurrentemente; la actividad de diseño está organizada en las siguientes fases:

1. **Diseño inicial**, que es la determinación de esquemas D y F armazón. Estas son representaciones iniciales de datos y funciones que pueden ser refinadas descendientemente en fases subsecuentes del diseño. En concreto, se debe dar prioridad al esquema F armazón y concentrar la atención en la descripción de entradas-salidas del sistema, identificando todas las interfaces y flujos de entrada-salida.
2. **Refinamientos progresivos descendentes**, realizados en paralelo para los esquemas D y F. El proceso de identificar los refinamientos de cada esquema está influido por las versiones previas de ambos. En este apartado, más adelante, se darán varias indicaciones para ejecutar tales refinamientos coordinados.
3. **Verificación de compleción mutua**, que se realiza frecuentemente durante el diseño conjunto (quizá después de cada refinamiento). Definimos la compleción mutua como sigue:
 - a) El esquema D está completo en relación con el esquema F si cada concepto expresado implícitamente en los flujos de datos y en los almacenes de datos del esquema F aparece en el esquema D. Para demostrar la compleción del esquema D, resulta muy útil considerar los esquemas exter-

nos y verificar que cada concepto mencionado en cualquiera de ellos esté también presente en el esquema D.

- b) El esquema F está completo respecto al esquema D si cada operación de recuperación o manipulación que deba ser realizada con los datos es en verdad realizada por un proceso del esquema F. En particular, para cada elemento de datos almacenado, debe haber un proceso encargado de su creación, y se espera que algún proceso se ocupe de su recuperación y eliminación.

4. Análisis final de los esquemas D y F: aquí se prueban las cualidades de los esquemas, y puede ser que se realice la reestructuración de algunos esquemas.

Este planteamiento descendente puro se puede modificar de acuerdo con las necesidades del diseñador; en particular, en el caso de aplicaciones complejas, los esquemas armazón pueden ser usados como punto de partida de una estrategia mixta que implique datos y funciones. En este caso, es esencial que cada par de refinamientos progresivos de datos y funciones se relacione con la misma vista de la aplicación. La integración final es guiada entonces por los esquemas armazón. Alternativamente, cada etapa de refinamiento puede llevarse a cabo con una estrategia diferente de la estrategia descendente pura; por ejemplo, los pasos del diseño funcional pueden realizarse con un planteamiento centrífugo.

En el curso de diseñar una aplicación particular, se puede destacar el diseño de datos o el diseño de funciones, dándole más realce a las decisiones de diseño producidas por cualquiera de ellos. Si se da más importancia a los datos o a las funciones, se dice que el diseño está **orientado a los datos u orientado a las funciones**, respectivamente. Así, la metodología de diseño conjunto de datos y funcional que se presenta en este capítulo representa una solución equilibrada dentro de un continuo cuyos extremos están representados por un diseño conceptual de datos puro o por un análisis funcional puro. Este planteamiento se demuestra por medio de un pequeño ejemplo de una oficina de registro; en el próximo capítulo se presenta un caso de estudio más grande. Considérese un sistema de información de estudiantes con los siguientes requerimientos.

La oficina de registro de una universidad tiene información sobre el plan de estudios de cada estudiante, donde se indican los cursos que éstos han seguido o van a seguir; también se registra cuándo completarán cada curso, información sobre la nota final, y la fecha de finalización. Para los estudiantes de doctorado, el departamento mantiene los nombres de los asesores y el título y la fecha del examen de grado. Esta información se modifica cuando el estudiante presenta su plan de estudio para el siguiente periodo, cuando los profesores entregan las calificaciones al final del periodo y cuando los estudiantes de doctorado escogen a sus asesores o definen sus exámenes de grado. Los estudiantes pueden solicitar información de sus calificaciones o de su avance en pos del doctorado.

Se empieza por proporcionar un esquema F armazón, que se muestra en la

parte superior de la figura 9.3a. La primera representación es bastante simple: se incluye un proceso único OFICINA_DE_REGISTRO; la interfaz ESTUDIANTE, que presenta solicitudes a la oficina (concernientes al registro de lista de estudios o al desarrollo del doctorado); la interfaz PROFESOR, que entrega las calificaciones, y el almacén de datos DATOS_DE_ESTUDIANTES. Teniendo en cuenta el esquema F, derivamos un esquema D armazón muy simple con las entidades ESTUDIANTE y CURSO, y la interrelación SEGUIDO_POR entre ellas.

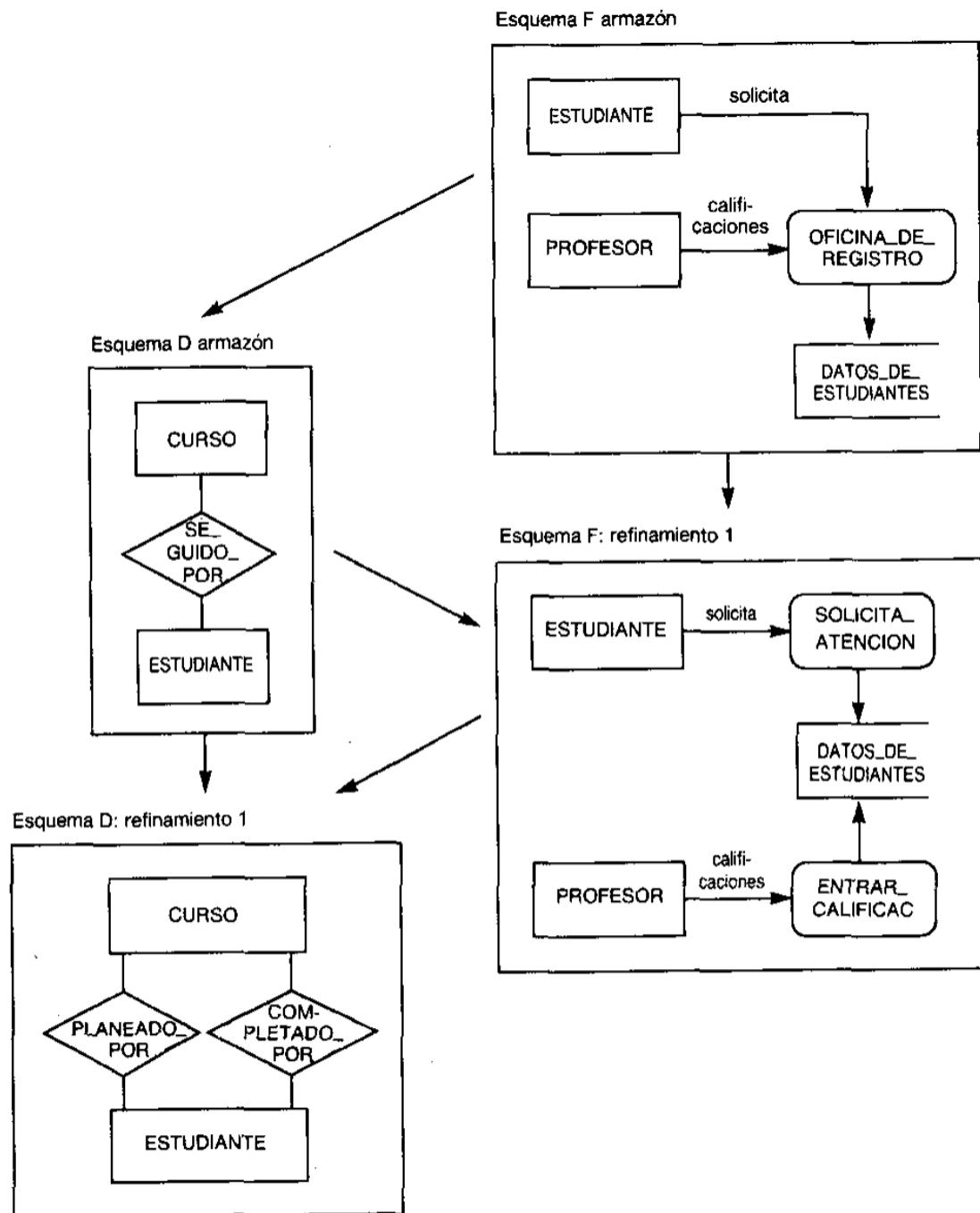
Ahora se considerará de nuevo el esquema F. Se refina el proceso OFICINA_DE_REGISTRO para dar dos procesos, ATENDER_SOLICITUDES e INTRODUCIR_CALIFICACIONES. Esto muestra, entre otras cosas, que la selección de cursos en el plan de estudio es obligación del estudiante, mientras que introducir las calificaciones es obligación de los profesores. Esto sugiere un refinamiento en el esquema D, que consiste en dividir la interrelación SEGUIDO_POR entre ESTUDIANTE y CURSO en las dos interrelaciones PLANEADO_POR (que indica todos los cursos insertados en el plan de cada estudiante) y COMPLETADO_POR (que indica los cursos que ya han sido completados por un estudiante, con la calificación final).

Se procede en seguida con el esquema F. El proceso ATENDER_SOLICITUDES se refina posteriormente para dar tres procesos, SELECCIONAR_SOLICITUD, PREPARAR_PLAN_DE_ESTUDIO, DIRIGIR_ESTUDIANTES_DE_DOCTORADO. De esta manera se han identificado los procesos para tratar los planes de estudio, los datos de los estudiantes de doctorado y las calificaciones de los cursos; se acepta este DFD como esquema F final; no se necesitan refinamientos posteriores.

A continuación se refina el esquema D. Como una consecuencia de la división del proceso ATENDER_SOLICITUDES, ahora se reconoce que algunas solicitudes son hechas por un subconjunto específico de estudiantes, e introducimos la entidad subconjunto ESTUDIANTES_DE_DOCTORADO. En seguida se refinan las entidades e interrelaciones añadiendo atributos. La interrelación PLANEADO_POR tiene un atributo OPCION_DEL_ESTUDIANTE (que puede ser una calificación de letra o aprobar/sin crédito); la interrelación COMPLETADO_POR tiene los atributos FECHA_FINAL y CALIFICACION. También se definen los atributos CODIGO, PROFESOR y NOMBRE para CURSO, ID_DEL_ESTUDIANTE y NOMBRE para ESTUDIANTE; y FECHA_ORAL, TITULO_DE_TESIS y ASESOR para ESTUDIANTES_DE_DOCTORADO. Se selecciona CODIGO como identificador de CURSO e ID_DEL_ESTUDIANTE como identificador de ESTUDIANTE.

De esta manera, se producen los esquemas D y F finales como resultado de dos refinamientos de los esquemas armazón iniciales, con influencia mutua. Los dos esquemas representan los requerimientos por completo y tienen las cualidades deseadas; en particular, la compleción mutua se alcanza porque todos los datos mencionados en los esquemas F son modelados en los esquemas D; a la inversa, todas las funciones requeridas para la manipulación de los datos del esquema D existen en el esquema F.

Las figuras 9.4a, b y c muestran los esquemas externos de los tres procesos PREPARAR_PLAN_DE_ESTUDIO, INTRODUCIR_CALIFICACIONES, DIRIGIR_ESTUDIANTES_DE_DOCTORADO. La compleción del esquema D puede ser compro-



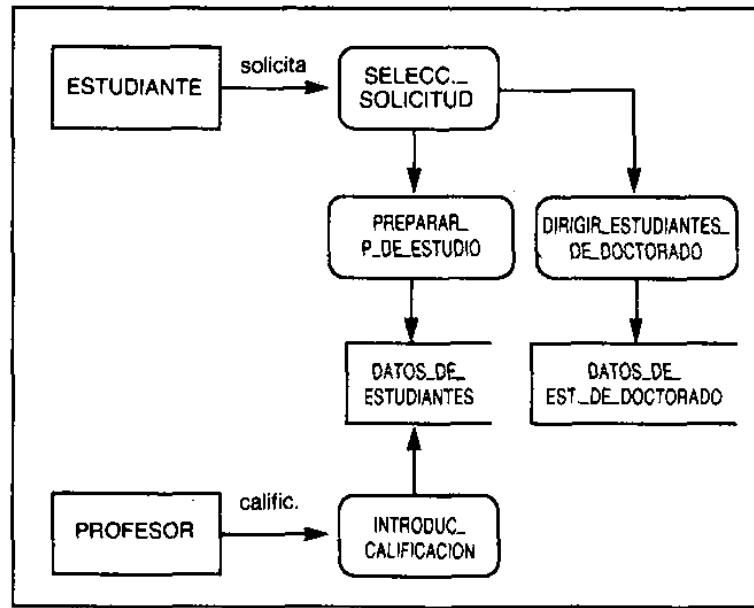
(a) Estructura de los esquemas D y F y primer refinamiento de ambos

Figura 9.3. Diseño conjunto de datos y funciones para el ejemplo de la oficina de registro.

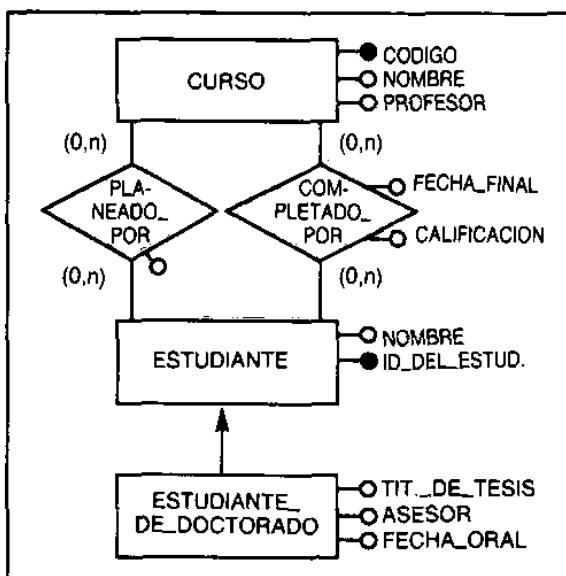
Esquema D: refinamiento 1

Esquema F: refinamiento 1

Esquema F: refinamiento 2 (esquema F final)



Esquema D: refinamiento 2 (esquema D final)



(b) Segundo refinamiento de los esquemas F y D

Figura 9.3.Bis Diseño conjunto de datos y funciones para el ejemplo de la oficina de registro (*continuación*).

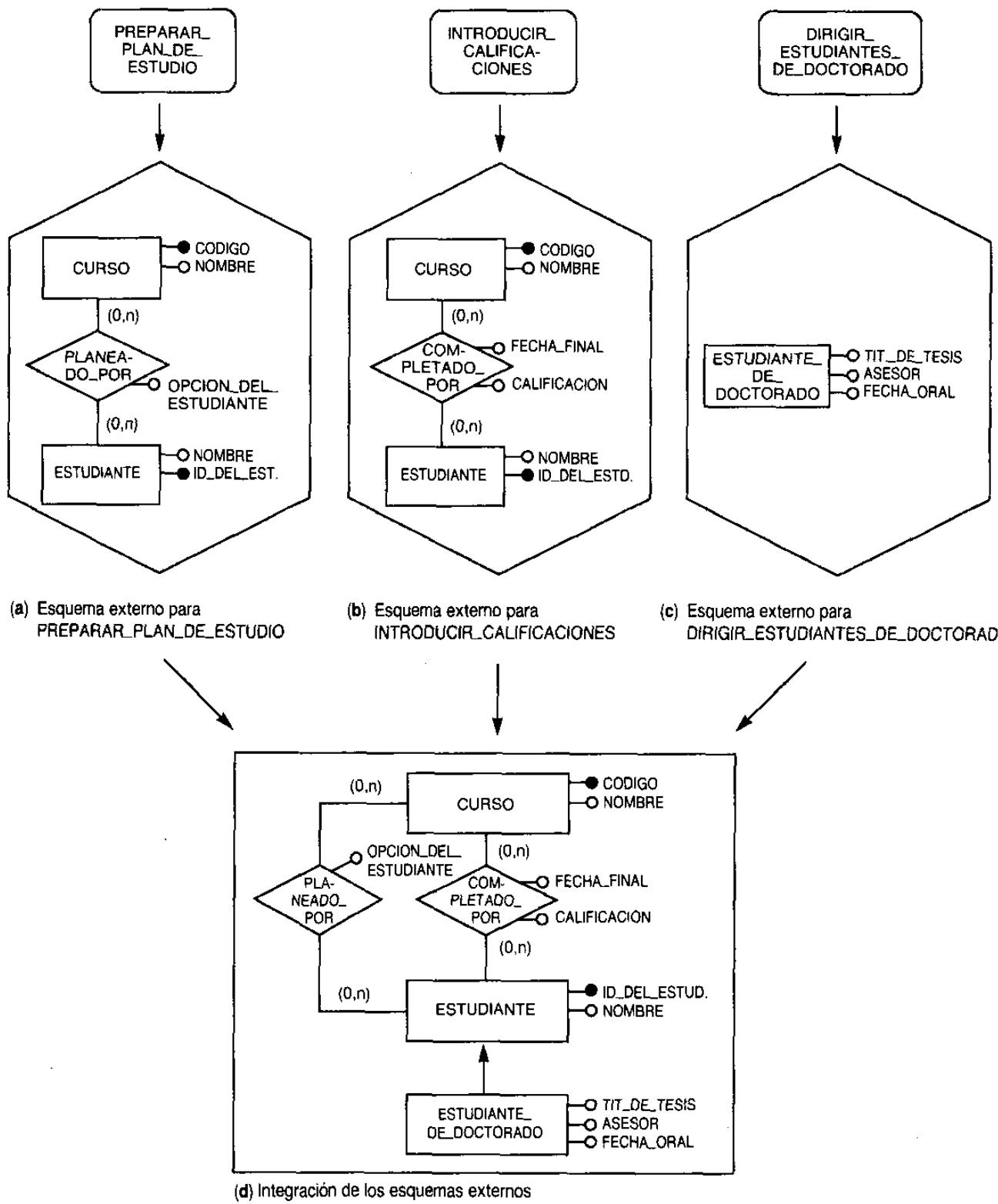


Figura 9.4. Verificación de compleción del esquema D usando esquemas externos.

bada verificando que todos los conceptos mencionados en el esquema externo estén representados en el esquema D-final. Efectivamente, al integrar los esquemas externos de los tres procesos se obtiene exactamente el esquema D final.

9.3. Sugerencias para refinamientos mutuos

La principal ventaja de diseñar los esquemas de datos y funciones conjuntamente es la posibilidad de alternar los refinamientos de cualquiera de los esquemas, de acuerdo con el estado actual de ambos. En este apartado se considerarán los cambios elementales que pueden ocurrir en cualquiera de los esquemas y se sugerirán los refinamientos que corresponden al otro.

Se considerará primero el caso en el cual se aplica una primitiva de refinamiento al esquema F. Supóngase que se aplica dicho refinamiento a un proceso P de un DFD y que EE es el esquema externo asociado con P; veremos cómo se puede hacer la modificación correspondiente a EE. Se supone que el esquema externo EE consiste inicialmente en una sola entidad E, y se refina P aplicando cualquiera de las primitivas T₁, T₂ o T₃ para la descomposición del proceso (Fig. 8.3), con lo que se generan dos procesos, P₁ y P₂. Se tienen los siguientes casos, que se ilustran en la figura 9.5:

1. Los dos procesos P₁ y P₂ se aplican a subconjuntos específicos de los casos de la entidad E. En este caso se puede refinar E para dar una jerarquía de generalización entre las tres entidades E, E₁, y E₂. Si E₁ y E₂ no comparten propiedades comunes, se puede refinar E para dar dos entidades separadas E₁ y E₂.
2. P₁ se aplica a todos los casos de E, pero P₂ se aplica sólo a un subconjunto de E. En este caso se puede refinar E, introduciendo una interrelación subconjunto entre E y una nueva entidad E₁.
3. Uno de los dos procesos, digamos P₁, opera sobre todos los casos de la entidad E, sin importar la generalización. En este caso el refinamiento del esquema F no sugiere un refinamiento correspondiente en el esquema externo.

Obsérvese que hablamos de refinamientos del esquema externo EE ; estos refinamientos serán transferidos después al esquema D. Cuando el refinamiento de P se hace usando la primitiva T₃, se coloca un nuevo almacén de datos entre P₁ y P₂. Este almacén de datos, por lo general, debe incluir un subconjunto del esquema externo de P; sin embargo, es también muy probable que tal primitiva pueda generar nuevos datos. Esto introduce una etapa ascendente en el desarrollo del esquema D.

Una situación similar ocurre cuando una entidad E del esquema D se refina a través de una jerarquía de generalización; supongamos, como antes, que el esquema externo EE del proceso P consiste sólo en la entidad E, pero ahora se deja que dicha entidad sea refinada a través de una generalización que conecta

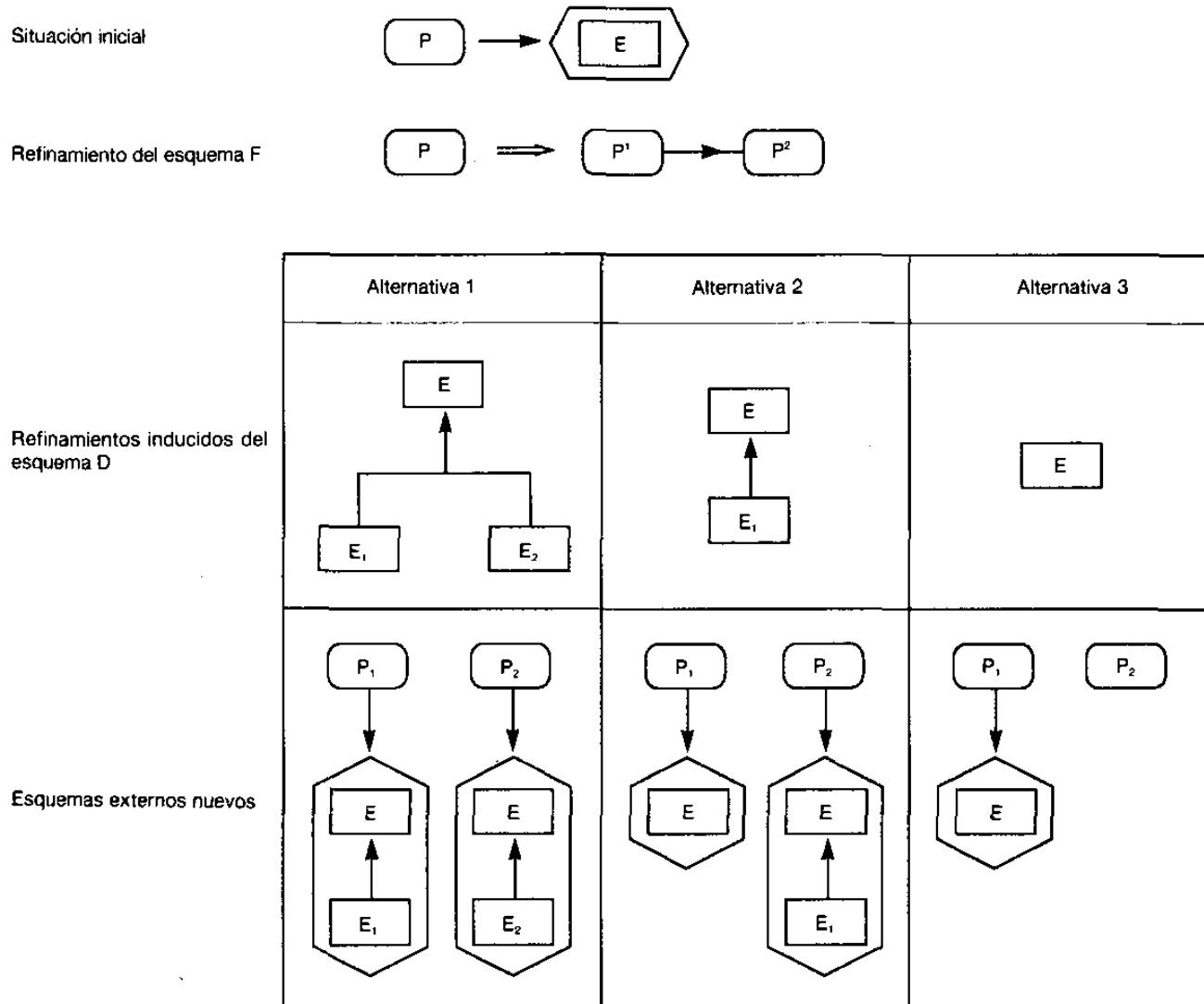


Figura 9.5. Ejemplos de refinamientos de datos inducidos por refinamientos de funciones.

E con las subentidades E_1 y E_2 . Los siguientes casos son entonces posibles, como se muestra en la figura 9.6:

1. El proceso P es también dividido en dos procesos, P_1 y P_2 , cada uno centrado en el uso de cualquiera de las nuevas subentidades. P_1 y P_2 pueden ser independientes (producidos por la primitiva T_3 , véase la figura 8.3) o estar conectados a través de un flujo (primitiva T_2) o un almacén de datos (primitiva T_1); los procesos están conectados cuando los casos de una de las dos subentidades, digamos E_1 , se usan antes que los datos de la otra sub entidad E_2 .

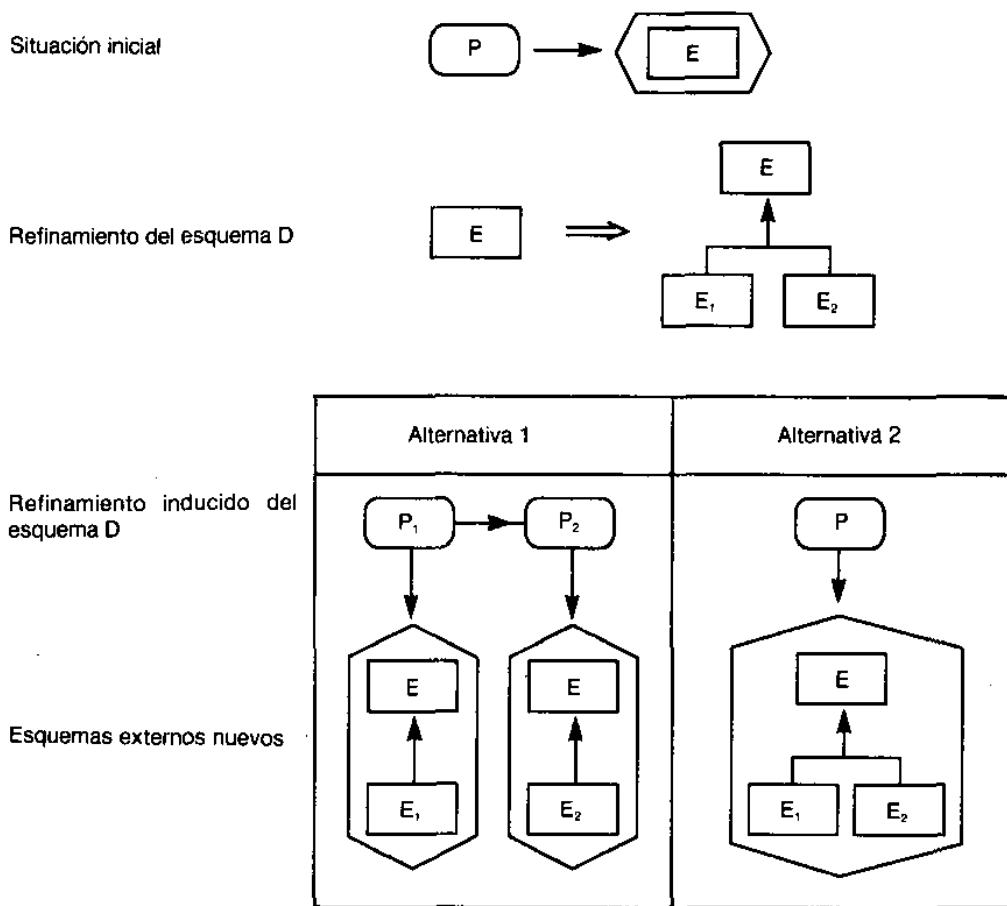


Figura 9.6. Ejemplo de refinamientos de funciones inducidos por refinamientos de datos.

2. El proceso P no necesita ser dividido, porque todos los casos de E, sin importar la existencia de subentidades, son usados por P.

Se puede considerar un esquema externo EE ligeramente más complejo, que consiste en dos entidades, **E₁** y **E₂**, conectadas por una interrelación R. En este caso P usa la interrelación R para tener acceso a una de las dos entidades, digamos **E₁**, partiendo de la otra. Entonces el refinamiento del proceso P para dar **P₁** y **P₂** puede indicar que la interrelación R debe ser refinada para dar **R₁** y **R₂** cuando los dos procesos, **P₁** y **P₂**, usen criterios diferentes para tener acceso a la entidad **E₁**. A la inversa, el refinamiento de R para dar **R₁** y **R₂** puede indicar que el proceso P debe ser refinado para dos procesos **P₁** y **P₂**, de manera que cada proceso use sólo una de las dos interrelaciones.

En el ejemplo de la oficina de registro, se usaron las transformaciones antes comentadas. Concretamente: 1) la división de la función **OFICINA_DE_REGISTRO** en dos funciones, **ATENDER_SOLICITUDES** e **INTRODUCIR_CALIFICACIONES**, que corresponden a la división de la interrelación **SEGUIDO_POR** en **PLANEADO_POR**,

Por cada PROCESO conectado con un ALMACEN DE DATOS:

1. Determinar las operaciones de base de datos, es decir, las interacciones individuales con la base de datos.
2. Para cada operación:
 - 2.1. Determinar el esquema de la operación, que es el subconjunto del esquema que contiene todos los datos mencionados por la operación.
 - 2.2. Determinar el esquema de navegación, que es un nuevo esquema con flechas que indican las condiciones de acceso y las navegaciones, y símbolos que indican las operaciones de acceso de base de datos realizadas (seleccionar, insertar, actualizar, suprimir).

Figura 9.7. Metodología para producir esquemas de navegación.

y COMPLETADO_POR, y 2) la generación de la función DIRIGIR_ESTUDIANTES_DE_DOCTORADO por la división de la función ATENDER_SOLICITUDES corresponde a la creación de la entidad subconjunto ESTUDIANTE_DE_DOCTORADO. Los dos refinamientos han sido inducidos en el esquema D después de una reestructuración del esquema F; en este sentido se puede decir que el ejemplo del apartado 9.2 se obtuvo emprendiendo acciones más decisivas sobre el esquema de funciones, es decir, con un enfoque (moderado) orientado a las funciones.

Conviene hacer hincapié en que los ejemplos que se consideran en este apartado son sólo los refinamientos típicos *sugeridos*; en general, los refinamientos reales son mucho más complejos. Los ejemplos nos dan un menú de patrones de refinamientos posibles que se pueden usar como marcos de referencia; el diseñador puede ser capaz de ajustarlos y adaptarlos al problema específico en consideración.

9.4. Esquemas de navegación para operaciones con bases de datos

En esta sección se introducen los conceptos nuevos de operación, esquema de operación y esquema de navegación, y luego se presenta una metodología progresiva para especificar esquemas de navegación de cada operación de base de datos. Esta metodología se ilustra en la figura 9.7. Derivar esquemas de navegación resulta útil para las fases subsecuentes del diseño de base de datos, en particular para el diseño lógico (descrito en la tercera parte).

La primera etapa consiste en identificar las operaciones de base de datos. Una operación de base de datos es una interacción elemental con la base de datos, que no incluye enunciados de control (por ejemplo, si-entonces-si no, mientras-hacer, repetir- hasta que) sino exclusivamente operaciones de recuperación o de actualización. Estas operaciones se pueden efectuar a través de una sola llamada a un sistema de base de datos (en el caso de un sistema relacional, a través de una sola consulta en SQL). La complejidad de la llamada puede variar, pero tarde o temprano resultará en la recuperación o actualización de algunos registros.

Las operaciones se realizan como parte de las actividades que tienen lugar

dentro de un proceso; en consecuencia, para identificar las operaciones se necesita detectar los procesos que las producen. Los procesos candidatos están directamente vinculados con algún almacén de datos. Se analizan entonces los procesos (complejos), para extraer de ellos varias operaciones (simples). En este capítulo no se considerará cómo interactúan las operaciones para construir aplicaciones complejas; se considerará, en consecuencia, cada operación de base de datos de forma independiente. Como quedará claro en los próximos capítulos, esta especificación es suficiente para el diseño lógico de la base de datos.

Para cada operación identificada se dibuja un **esquema de operación**, que consiste en el subconjunto del esquema externo del proceso que utiliza realmente la operación. El esquema de operación contiene todos los elementos del esquema externo mencionados en el esquema de especificación o requeridos para la navegación, en caso de que no estén explícitamente mencionados. Si no se cuenta con el esquema externo del proceso, el esquema de operación debe ser extraído del esquema D completo. Un esquema de operación incluye: 1) atributos que se usan para las condiciones que gobiernan el acceso a las entidades e interrelaciones, o su selección; 2) atributos que usan las operaciones de recuperación o actualización; 3) entidades o interrelaciones que son insertadas o suprimidas; y 4) interrelaciones o jerarquías de generalización que se usan para la navegación.

El esquema de la operación es una estructura plana que no representa cómo se realiza el procesamiento de la base de datos; de hecho, una operación usa los conceptos de la base de datos en una secuencia dada, proporcionando las condiciones de acceso (predicados de selección), especificando navegaciones por las interrelaciones; y finalmente ejecutando acciones de lectura, modificación, eliminación o inserción sobre las entidades o interrelaciones seleccionadas. Tal secuencia de acciones está representada en un esquema de navegación, que es una modificación del **esquema de operación** previo que incluye símbolos especiales: 1) las flechas que apuntan a los atributos indican atributos usados en las condiciones de selección; 2) las flechas entre las interrelaciones o generalizaciones indican navegación; y 3) los símbolos R, M, I, y D dentro de las entidades o interrelaciones indican respectivamente las acciones leer, modificar, insertar y suprimir.

Adviértase que la misma entidad o interrelación puede ser usada de diferentes maneras por la misma operación y esto puede derivar en una repetición de la entidad o interrelación en el esquema de navegación. De hecho, después de repetirse las entidades e interrelaciones, los ciclos dentro del esquema de operación se eliminan y, consecuentemente, las entidades e interrelaciones quedan parcialmente ordenadas por la relación de precedencia asociada con las flechas tipo 2; en la mayoría de los casos, este orden parcial se reduce a una secuencia simple o a un árbol. Obsérvese que los esquemas de navegación son un medio de indicar gráficamente qué es lo que está implicado en una operación de alto nivel; no decimos que la notación del esquema de navegación sea completa o correcta. Es decir, no podemos afirmar que la notación sea adecuada para representar todas las operaciones.

Proceso P₁: INTRODUCIR_LAS_CALIFICACIONES_DE_UN_CURSO

- O₁ : Introducir las calificaciones de un curso
 O₂ : Cambiar las calificaciones de un curso
 O₃ : Determinar la curva de calificaciones de un curso y calcular su desviación con respecto a una curva normal.

Proceso P₂: PREPARAR_PLAN_DE_ESTUDIO

- O₄ : Insertar o suprimir un estudiante
 O₅ : Introducir el plan de estudio de un estudiante (para un semestre en particular)
 O₆ : Cambiar la opción del estudiante para un curso
 O₇ : Retirar una materia del plan de estudio de un estudiante

Proceso P₃: DIRIGIR_ESTUDIANTES_DE_DOCTORADO

- O₈ : Introducir un nuevo estudiante de doctorado y su asesor
 O₉ : Introducir el título de la tesis y la fecha del examen oral
 O₁₀: Detectar todos los estudiantes que han recibido una B o menos en un curso impartido por un profesor dado

Figura 9.8. Operaciones para el ejemplo de la oficina de registro.

La metodología antes mencionada se aplicará en el ejemplo de la oficina de registro que se introdujo en el apartado anterior. Una lista de las operaciones se ofrece en la figura 9.8; la lista se explica por sí misma con la excepción de la operación O₁₀, que será desarrollada posteriormente en el resto de este apartado. La operación O₁₀ se especifica como sigue: «Detectar todos los estudiantes que han recibido una B o menos en un curso impartido por un profesor dado». El esquema de operación correspondiente se muestra en la figura 9.9 e incluye los atributos PROFESOR, CALIFICACION, y NOMBRE, extraídos respectivamente de CURSO, COMPLETADO_POR, y ESTUDIANTE.

Este esquema da lugar al esquema de navegación que se muestra en la figura 9.10. Respecto al esquema de operación, se añaden las siguientes características: 1) la anotación R está presente en todos los conceptos de los esquemas para in-

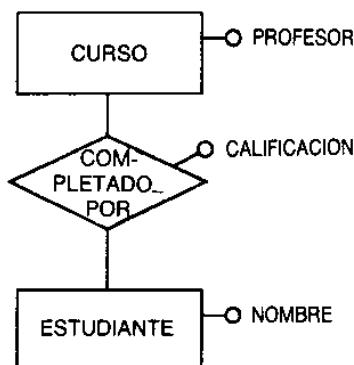


Figura 9.9. Ejemplo de un esquema de operación.

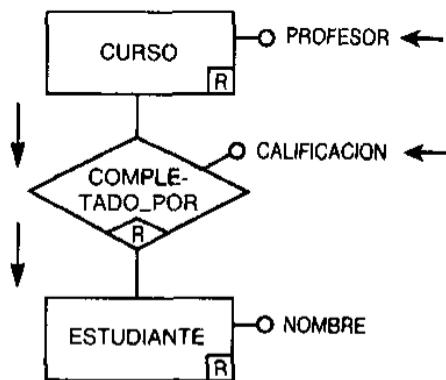


Figura 9.10. Ejemplo de esquema de navegación.

dicar que se usan para la recuperación; 2) hay flechas que apuntan a los atributos CALIFICACIONES y PROFESOR, utilizados para las condiciones de recuperación; y 3) las flechas a lo largo de la interrelación COMPLETADO_POR que indican que la interrelación se recorre de CURSO a ESTUDIANTE. Obsérvese que el tercer punto arriba mencionado se debe al juicio subjetivo del diseñador. La flecha puede dibujarse en cualquier sentido cuando hay condiciones en los dos lados de la interrelación.

Otros ejemplos de esquemas de operación y navegación se construyen a partir del esquema D en la figura 9.11 (véase el ejercicio 9.6). Considérese la operación: «Seleccionar los nombres de los médicos que viven en la misma ciudad donde nacieron». El esquema de operación se muestra en la figura 9.12; incluye todas las interrelaciones, las entidades PERSONA y CIUDAD, y los atributos NOMBRE y PROFESION de PERSONA y NOMBRE de CIUDAD. El esquema de navegación se muestra en la figura 9.13, y presenta una estructura notablemente diferente; indica que la condición inicial está en PROFESION de PERSONA, para seleccionar

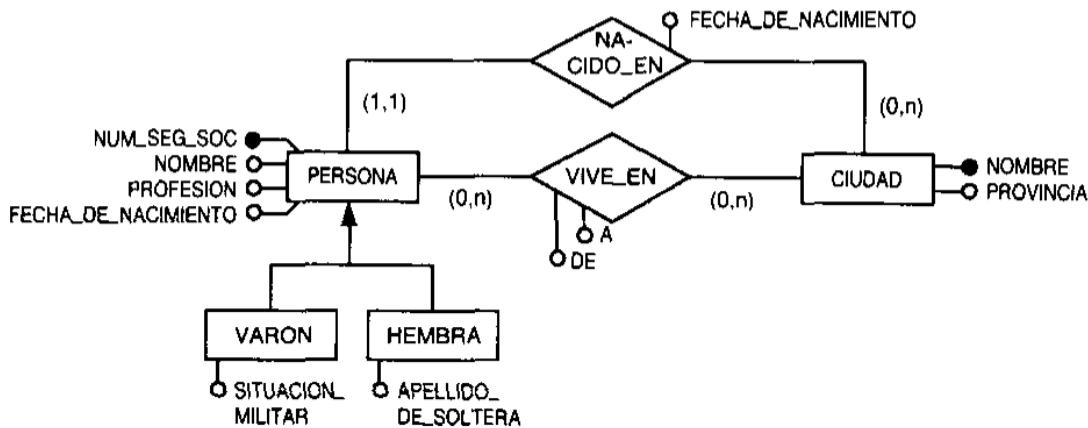


Figura 9.11. Esquema de datos para exemplificar esquemas de navegación.

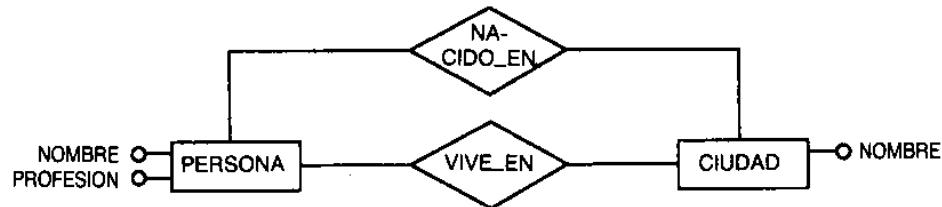


Figura 9.12. Ejemplo de esquema de operación.

aquellas personas que son médicos; entonces, se recorren las interrelaciones NA-CIDO_EN y VIVE_EN para seleccionar dos casos de CIUDAD. Si estas ciudades tienen el mismo nombre, la persona seleccionada satisface todas las condiciones, y el nombre puede ser recuperado. Así, el esquema de navegación representa una posible manera (entre otras) de derivar los nombres de los médicos que cumplen todas las condiciones. Se proporcionarán muchos otros ejemplos de esquemas de navegación al final del siguiente capítulo.

9.5. Resumen

En este capítulo se ha mostrado una metodología para vincular los esquemas de datos, denominados *esquemas D*, y los de funciones, denominados *esquemas F*, a través de esquemas externos, refinamientos mutuos y esquemas de navegación, para producir una especificación conceptual conjunta de datos y funciones. Este proceso es similar al de un escalador de montañas, que alterna los movimientos de los miembros izquierdos y derechos. El número de movimientos y su selección para un proceso de diseño específico son altamente subjetivos, pero afirmamos que este enfoque facilita el entendimiento progresivo de la naturaleza de los datos y funciones, así como la comparación cruzada para lograr compleción y coherencia.

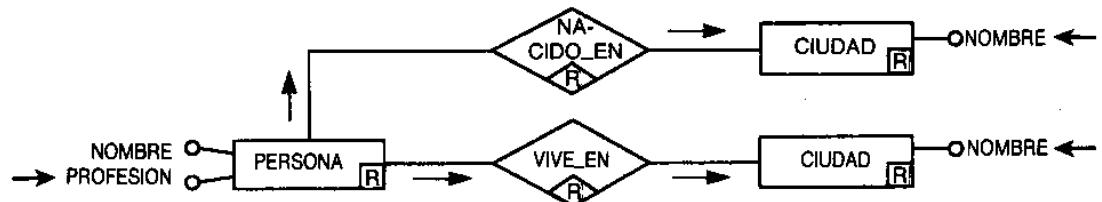


Figura 9.13. Ejemplo de esquema de navegación.

Ejercicios

- 9.1. Diseñe los esquemas externos para el DFD de fútbol que se muestra en la figura 8.12; haga suposiciones explícitas cuando se necesite.
- 9.2. Explique la diferencia entre los esquemas externos usados en las arquitecturas de base de datos (por ejemplo, las tres arquitecturas de esquemas) y los introducidos en este capítulo.
- 9.3. Considere el ejemplo de la oficina de registro; proponga un desarrollo diferente de este caso de estudio, en el que los refinamientos para este esquema D induzcan refinamientos en el esquema F.
- 9.4. Aplique la metodología de diseño conjunto de datos y funciones al siguiente caso de estudio:
En un amplio valle alpino bien comunicado, todas las sillas para esquiadores y teleféricos están sujetos a un sistema único de billetes, aunque pertenezcan a distintos propietarios. Los billetes tienen una banda magnética que puede ser leída por un dispositivo magnético cuando cada esquiador espera su turno para subir. Los dispositivos magnéticos están conectados a un computador central, que mantiene información sobre el uso de las sillas y teleféricos. Cada semana los beneficios de la venta de billetes se distribuyen entre los propietarios, proporcionalmente al uso. Los esquiadores son identificados por un número único, y se conservan datos estadísticos en relación a cuántas veces cada silla o teleférico es usado por los esquiadores que han comprado su billete en una taquilla específica. Esta información se usa para determinar lo lejos que viaja cada esquiador en el valle, suponiendo que los esquiadores compran las billetes en un lugar cercano a sus residencias.
- 9.5. Derive los esquemas de operación y navegación para las tres operaciones del caso de estudio del ejercicio 9.4.
- 9.6. Traduzca a esquemas de navegación, usando transformaciones paso por paso, las siguientes operaciones con esquema de la figura 9.11.
 - a) Recupere los nombres y direcciones de todos los programadores que viven en ciudades con menos de 1000 habitantes.
 - b) Recupere los nombres y direcciones de todos los programadores que viven en Milán y tienen más de 50 años de edad.
 - c) Inserte el hecho de que una mujer determinada, con cierto número de seguro social, tiene un apellido de soltera que anteriormente se dejó sin especificar.
 - d) Inserte la información de que el capitán John Dow nació en Minneapolis en 1955.
 - e) Recupere el número de habitantes de las ciudades en las que vive una alcaldesa.

Bibliografía

T.W. Olle, H.G. Sol, y A.A. Verrijn-Stuart, eds., *Information Systems Design Methodologies: A Comparative Review*, North-Holland, 1982 (conferencia CRIS 1).

T.W. Olle, H.G. Sol, y C.J. Tully, eds., *Information Systems Design Methodologies: A Feature Analysis*, North-Holland, 1983 (conferencia CRIS 2).

Estos dos libros presentan una comparación de varias metodologías para análisis de datos y funcional. Todas las metodologías se ven trabajando con el mismo caso de estudio, la organización de una conferencia de trabajo IFIP. El primer libro presenta la aplicación de siete metodologías al caso de estudio; el segundo presenta varios trabajos de repaso que comparan esas metodologías detalladamente.

J.L. Carswell, jr. y S.B. Navathe, «SA-ER: A Methodology That Links Structured Analysis and Entity-Relationship Modeling for Database Design». En S. Spaccapietra, ed., *Proc. Fifth International Conference on Entity-Relationship Approach*, Dijon, Francia, North-Holland, 1986.

Este trabajo presenta un enfoque metodológico para el análisis y diseño de sistemas de información e integra diagramas de flujo de datos y esquemas ER. En el enfoque propuesto, los DFD son modelados inicialmente, uno para cada «unidad» del sistema de información; luego, para cada unidad, se produce una representación ER de los datos. Así, después de estas actividades de modelado, el sistema de información se describe por medio de varios DFD, producidos con un planteamiento descendente, y varios esquemas de datos ER independientes. Al final, todos los esquemas de datos se integran en uno solo, usando las técnicas de integración de esquemas presentadas en el capítulo 5, con un planteamiento ascendente.

A. Cornelio y S.B. Navathe, «Database Support for Engineering CAD and Simulation». En *Proc. Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, ACM-SIGMOD, IEEE, NIST, Gaithersburg, Md., 1989.

A. Cornelio y S.B. Navathe, «Modeling Engineering Data By Complex Structural Objects and Complex Functional Objects». En *Proc. International Conference on Extending Data Base Technology*, Venecia, Italia, 1990.

A. Cornelio, S.B. Navathe y K.L. Doty, «Extending Object-Oriented Concepts to Support Engineering Applications». En *Proc. IEEE Sixth International Conference on Data Engineering*, Los Angeles, IEEE, 1990.

S.B. Navathe y A. Cornelio, «Database Integration for Engineering Design». *Proc. First International Conference on System Integration*, Morristown, N.J., IEEE, 1990.

Estos trabajos presentan un planteamiento para modelar entornos complejos, sistemas físicos, sistemas de ingeniería, o cualquier entorno similar por medio del *paradigma estructura función* (S-F). Este enfoque es una extensión del planteamiento de modelado orientado a objetos para lograr una mejor caracterización de las interacciones complejas y de la dinámica de los sistemas complejos. En el paradigma S-F, estructura y función se representan usando distintos tipos de objeto; la correspondencia general de muchos a muchos entre estructuras y funciones se capta mediante un tercer tipo de objeto, denominado *tipo de objeto de interacción*. Las reglas para abstraer estructuras y funciones para agregaciones y generalizaciones se desarrollan detalladamente.

Estudio de un caso

En este capítulo se presenta un amplio caso de estudio: se aplica la metodología de diseño conjunto de datos y funcional al sistema de información de una compañía de autobuses. Primero se presentan los requerimientos del caso de estudio y luego se desarrollan los esquemas F y D armazón. Cada esquema se refina entonces dos veces para lograr los esquemas D y F finales. Se verifica la compleción mutua y después se desarrollan los esquemas de navegación para las operaciones más importantes. Este caso de estudio es también la base de los ejemplos de la tercera parte del libro.

10.1. Requerimientos

La compañía está dividida en dos grandes áreas: *gestión de pasajeros* y *gestión de servicios*. La gestión de pasajeros se encarga de las reservas de asientos de autobús y de dar información a los pasajeros y agencias de viajes sobre horarios y tarifas. La gestión de servicios organiza los viajes ordinarios y especiales. Se omite la descripción de las oficinas de contabilidad y de personal, que también forman parte de la compañía de autobuses, para que el caso de estudio sea más manejable. En particular, no se tratarán ni la venta de billetes, ni la contabilidad de las ventas después de los viajes.

10.1.1. Gestión de pasajeros

Los pasajeros o las agencias de viajes hacen las reservas de los asientos directamente. Pueden hacerlas para cualquier servicio de autobús (ordinario o especial) y son gratuitas. Los pasajeros pueden pedir área de fumadores o de no fumadores. Se puede hacer reservas para segmentos del viaje, así que el mismo asiento puede estar disponible para diferentes personas en diferentes partes del mismo viaje. La oficina guarda *las hojas de descripción de viaje*, que se preparan con cuatro semanas de antelación a la fecha del viaje; éstas incluyen el horario del viaje, la lista de todas las paradas intermedias y espacios para cada

asiento, donde es posible anotar las reservas. La oficina de gestión de servicios prepara las hojas de descripción de viaje y las usa *la oficina de reservas* para registrar las reservas. Son devueltas a la oficina de gestión de servicios al menos dos horas antes de la salida; por ello no se puede hacer reservas inmediatamente antes de la salida.

La oficina principal tiene un despacho y algunas líneas de teléfono exclusivas para las reservas. La información sobre las tarifas se da en una oficina diferente, denominada *oficina de tarifas*, que también cuenta con varias líneas de teléfono exclusivas. Las tarifas se informan en un *folleto de descripción de tarifas*, que se prepara mensualmente; y son relativamente estables. Los empleados y operadores de cualquiera de las líneas telefónicas no pueden contestar preguntas que conciernen a otra oficina; a lo sumo, pueden mandar al cliente al otro despacho o transferir las llamadas de teléfono. Así, se distinguen dos oficinas separadas, para reservas y para tarifas.

Normalmente, los pasajeros también solicitan información sobre el *horario* de los viajes. Esta información la prepara la oficina de gestión de servicios, pero es mucho más variable; los horarios se pueden modificar dependiendo de las condiciones de la carretera o de sucesos especiales. Se pueden organizar viajes extraordinarios para eventos especiales. Así, la reunión de información sobre horarios se centraliza en una tercera oficina, denominada *oficina de horarios*, que apoya a las otras dos oficinas. Las preguntas de los pasajeros sobre horarios se pueden contestar directamente en la oficina de reservas y en la de tarifas, porque los empleados de estas oficinas pueden usar los horarios o las hojas de descripción de viaje. Las modificaciones al horario de los viajes se comunican a la oficina de horarios, que es la responsable de actualizar las hojas de descripción de los viajes (y algunas veces de comunicar los cambios a los pasajeros que tienen reservas).

Los pasajeros se pueden inscribir en un programa de *viajero frecuente*, y los que lo hacen ganan un premio cuando acumulan un número determinado de kilómetros.

10.1.2. Gestión de servicios

Se distinguen cuatro oficinas principales dentro del área de gestión de servicios, que se ocupan de los viajes ordinarios, los viajes especiales, la gestión de los autobuses y la gestión de los conductores de los autobuses. La oficina de gestión de *viajes ordinarios* prepara las hojas de descripción de los viajes y el folleto de descripción de tarifas. En cada temporada se reorganizan los viajes ordinarios para cumplir con los distintos requerimientos de los viajeros. Los oficiales del servicio de mantenimiento de carreteras y del gobierno dan información que se usa para modificar los viajes.

Algunas veces, a causa de sucesos externos (como partidos de fútbol, elecciones, ferias, días festivos, etc.), la compañía decide programar viajes especiales; cuando sucede esto, la oficina de gestión de *viajes especiales* prepara una

hoja de descripción de viaje. Tiene entonces la obligación de comunicar la información sobre los viajes especiales a las oficinas de tarifas y de reservas.

La oficina de *gestión de autobuses* tiene el objetivo de organizar cada viaje individual. Específicamente, recibe las hojas de descripción de viaje con las reservas de los pasajeros anotadas en ellas y añade el nombre del conductor del autobús, la marca y el número de matrícula del autobús y el número máximo de pasajeros que puede aceptar. También se comunican las tarifas para cada segmento del viaje; de manera que toda la información que necesita el conductor del autobús está registrada en la hoja de descripción de viaje. Otras actividades de la gestión de autobuses son resolver los problemas mecánicos de los autobuses, su compra y venta, reparar los daños, etc. La oficina guarda los documentos de los autobuses y un archivo (en papel) de cada autobús en el que se registran las reparaciones, mantenimiento y uso del autobús.

La oficina de *gestión de conductores* guarda un archivo con los datos individuales de cada conductor. Cuando un conductor enferma, comunica su ausencia a la oficina. Igualmente, el conductor comunica a esta oficina todos los problemas mecánicos que descubre en su autobús. En la práctica, esta oficina provee de asistencia a los conductores y los controla.

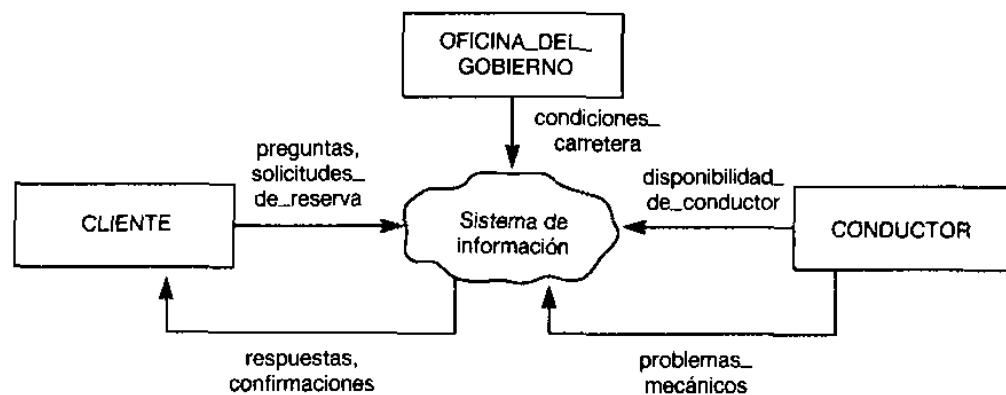
10.2. Esquemas de armazón

Inicialmente, producimos un esquema F armazón. De acuerdo con la metodología de la figura 9.2, primero identificamos las siguientes interfaces:

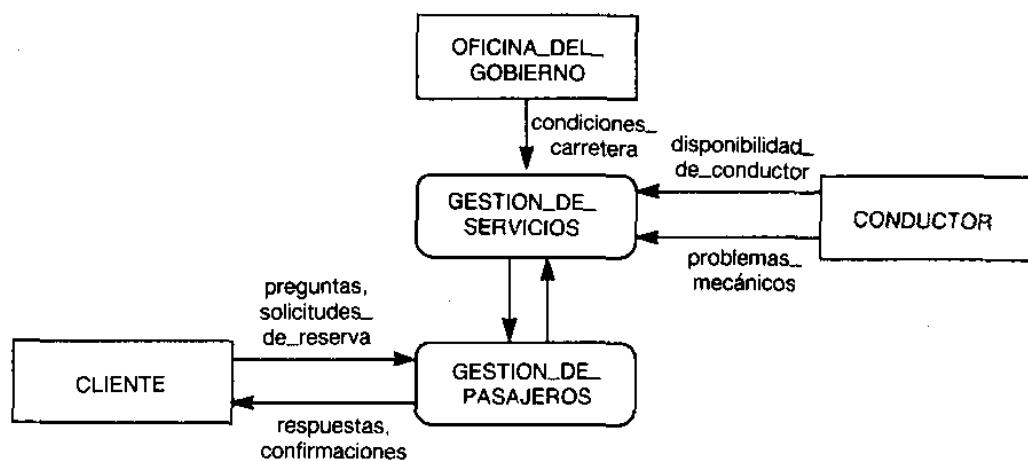
1. CLIENTE, esto es, personas y agencias de viajes que necesitan información y hacen reservas.
2. OFICINA_DEL_GOBIERNO, oficinas que interactúan con el sistema comunicando las condiciones de las carreteras y solicitando viajes especiales.
3. CONDUCTOR, que representa los conductores de los autobuses, los cuales comunican al sistema su disponibilidad y el estado de los autobuses.

El resultado de este primer análisis de actividad es el esquema de *caja negra* de la figura 10.1a. Se refina entonces el esquema F, identificando el proceso principal del sistema y los flujos entre ellos. Una primera descomposición es inmediatamente sugerida por la descripción de los requerimientos, donde se distinguen las dos oficinas principales, GESTION_DE_PASAJEROS y GESTION_DE_SERVICIOS. Por tanto, los flujos se refinan como sigue:

1. CLIENTE interactúa solamente con GESTION_DE_PASAJEROS.
2. OFICINA_DEL_GOBIERNO y CONDUCTOR interactúan sólo con GESTION_DE_SERVICIOS.
3. Dos flujos, que se dejan sin especificar, describen el intercambio de información entre GESTION_DE_PASAJEROS y GESTION_DE_SERVICIOS.



(a) Esquema F de caja negra



(b) Esquema F armazón completo

Figura 10.1. Esquema F armazón.

De esta manera se completa el esquema F armazón.

Ahora consideraremos los esquemas externos de los dos procesos GESTION_DE_SERVICIOS y GESTION_DE_PASAJEROS, que se muestran en la figura 10.2. El primero tiene simplemente las entidades VIAJE y CONDUCTOR y la interrelación CONDUCIDO POR entre ellas; el segundo tiene las entidades VIAJE y CLIENTE y la interrelación HACE entre ellas.

Al derivar un primer esquema D se puede aprovechar la información previa expresada en el esquema F. Comenzando con las interfaces, está claro que CLIENTE es un concepto del esquema D, ya que la información del cliente es esencial para la actividad de reserva. Igualmente, CONDUCTOR es otro concepto del esquema D, ya que hay una mención explícita de la gestión de la información de los conductores. Por el contrario, las oficinas del gobierno no desem-

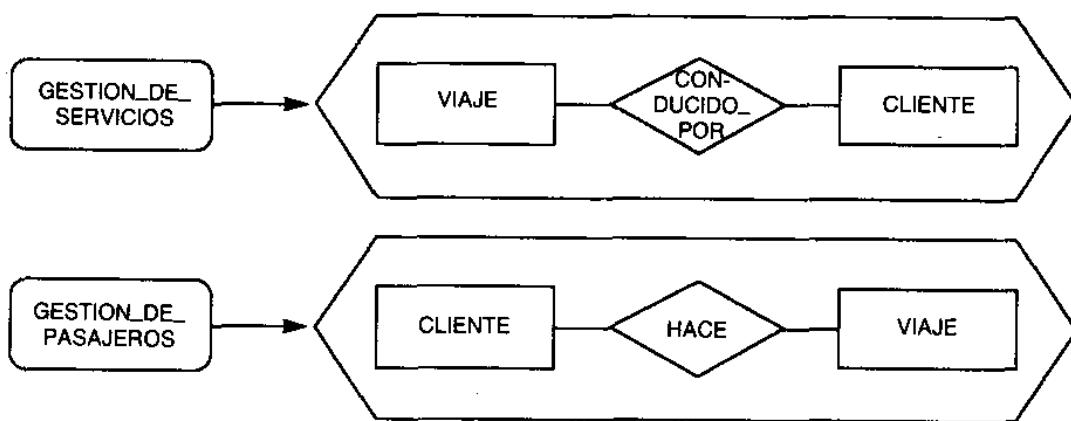


Figura 10.2. Esquemas externos para GESTION_DE_PASAJEROS y GESTION_DE_SERVICIOS.

peñan ningún papel específico dentro del sistema de información; son sólo fuentes de datos. Obsérvese que la selección de conceptos en el esquema D ayuda a fijar las fronteras del dominio de aplicación, y que esta selección es muchas veces difícil y arbitraria. Se modela CLIENTE y CONDUCTOR como entidades. Estas no están directamente conectadas, pero ambas están conectadas al mismo concepto, VIAJE, que también está modelado como entidad. Así, el esquema D preliminar incluye las entidades CLIENTE, CONDUCTOR, y VIAJE, y las interrelaciones HACE (entre CLIENTE y VIAJE) y CONDUCIDO_POR (entre VIAJE y CONDUCTOR). El esquema D correspondiente se muestra en la figura 10.3a. Obsér-

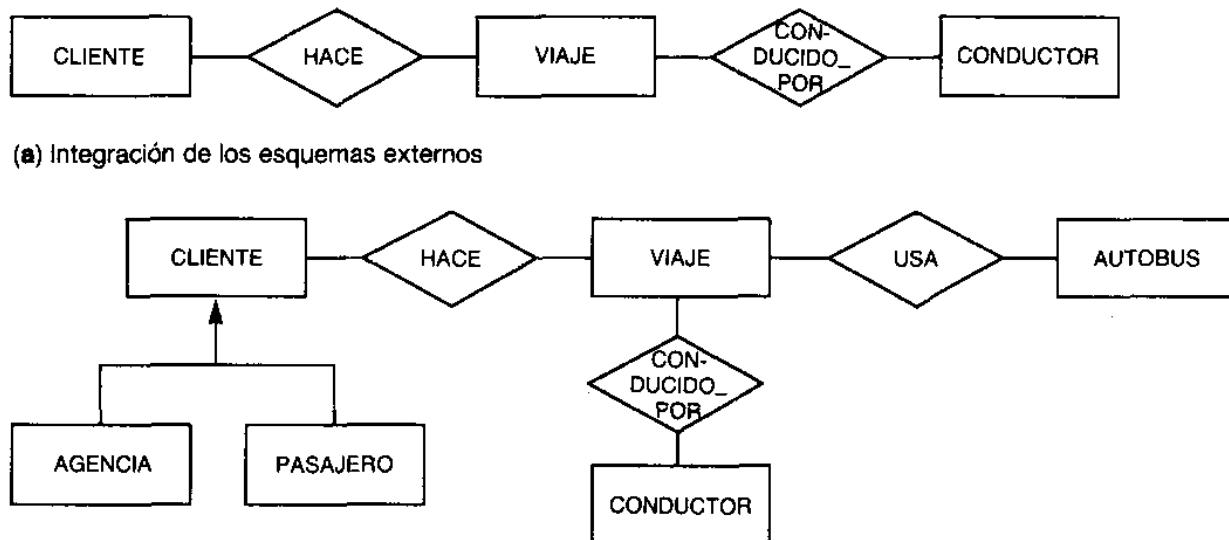


Figura 10.3. Esquema D armazón.

vese que este esquema se obtiene al integrar los dos esquemas externos de la figura 10.2.

Ahora refinamos este esquema. El primer refinamiento se aplica a la entidad CLIENTE; se introduce una generalización con las dos entidades PASAJERO y AGENCIA. Como los clientes están repartidos entre pasajeros y agencias, esta generalización es total y exclusiva. Se postula que las agencias, a su vez, reservan los asientos para sus clientes, aunque esto está fuera del ámbito del caso de estudio.

Se introduce entonces AUTOBUS, una entidad ausente que claramente debe pertenecer al esquema D; se conecta VIAJE con AUTOBUS a través de la interrelación USA. En este punto se tiene un esquema en el que todas las entidades e interrelaciones representan conceptos aproximadamente en el mismo nivel de abstracción; éste es un buen esquema D armazón que capta la esencia de nuestra aplicación, como se muestra en la figura 10.3b.

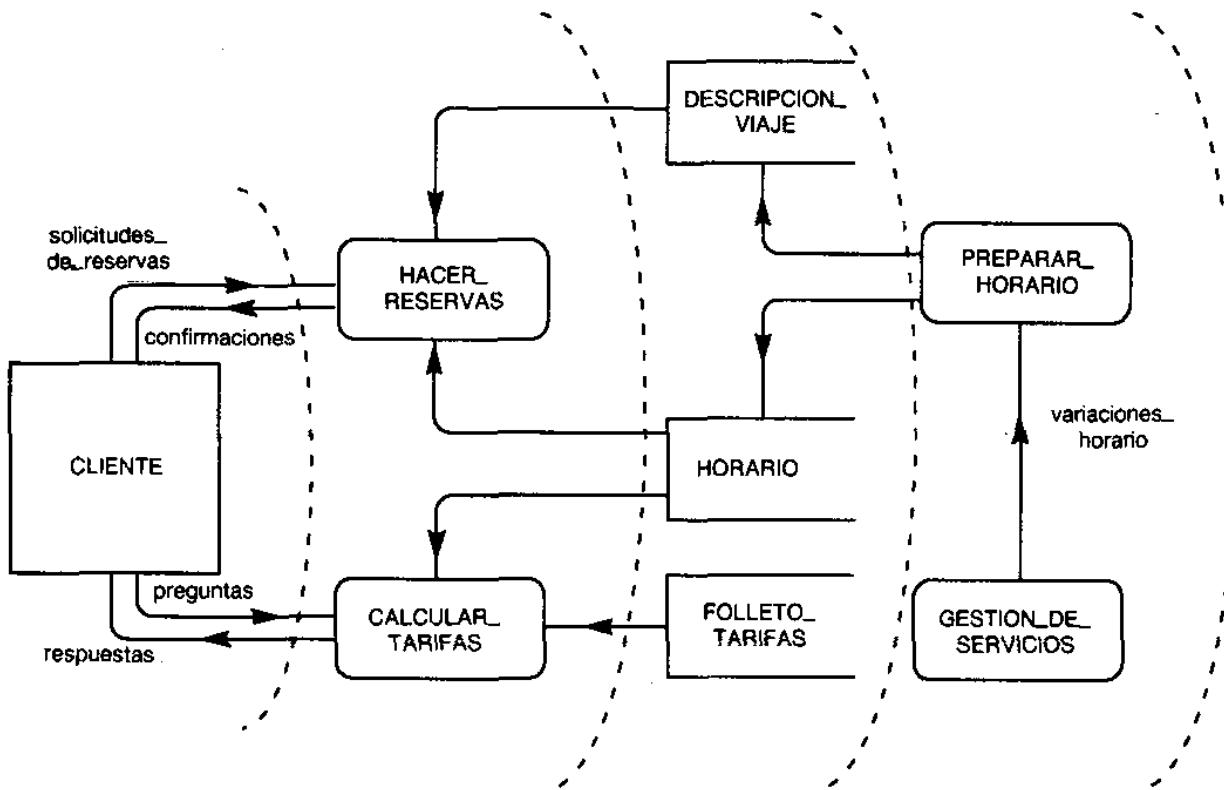
10.3. Primer refinamiento

Se refinará primero el esquema F, y luego el esquema D. Una vez más se usan los esquemas externos como guía para el refinamiento del esquema D.

10.3.1. Esquema F

En el refinamiento del esquema F, se puede proceder refinando los dos procesos GESTION_DE_SERVICIOS y GESTION_DE_PASAJEROS independientemente. Se considerará primero GESTION_DE_SERVICIOS. Nos concentraremos en la interfaz CLIENTE y usaremos una estrategia centrífuga siguiendo los flujos de datos, descubriendo al mismo tiempo procesos y almacenes nuevos. CLIENTE puede interactuar con dos oficinas diferentes que corresponden a procesos distintos: HACER_RESERVAS y CALCULAR_TARIFAS. Las solicitudes de reservas y las confirmaciones se intercambian entre CLIENTE y HACER_RESERVAS las preguntas y respuestas acerca de las tarifas se intercambian entre CLIENTE y CALCULAR_TARIFAS. En seguida se determinan los almacenes de datos que se requieren para cada proceso: 1) HORARIO, disponible tanto para HACER_RESERVAS como para CALCULAR_TARIFAS; 2) DESCRIPCION_VIAJE, disponible para HACER_RESERVAS, y 3) FOLLETO_TARIFAS, disponible para CALCULAR_TARIFAS.

Continuando el proceso centrífugo, se descubre que los horarios pueden ser modificados por otro proceso de la oficina de gestión de pasajeros, denominado PREPARAR_HORARIO. Este proceso, a su vez, recibe un flujo de información acerca de las variaciones de los horarios de los viajes procedente de la oficina de gestión de servicios. Se omite mostrar en este DFD cómo se preparan DESCRIPCION_VIAJE, HORARIO y FOLLETO_TARIFAS, dejando esto para futuros refinamientos. El esquema F correspondiente se muestra en la figura 10.4a.

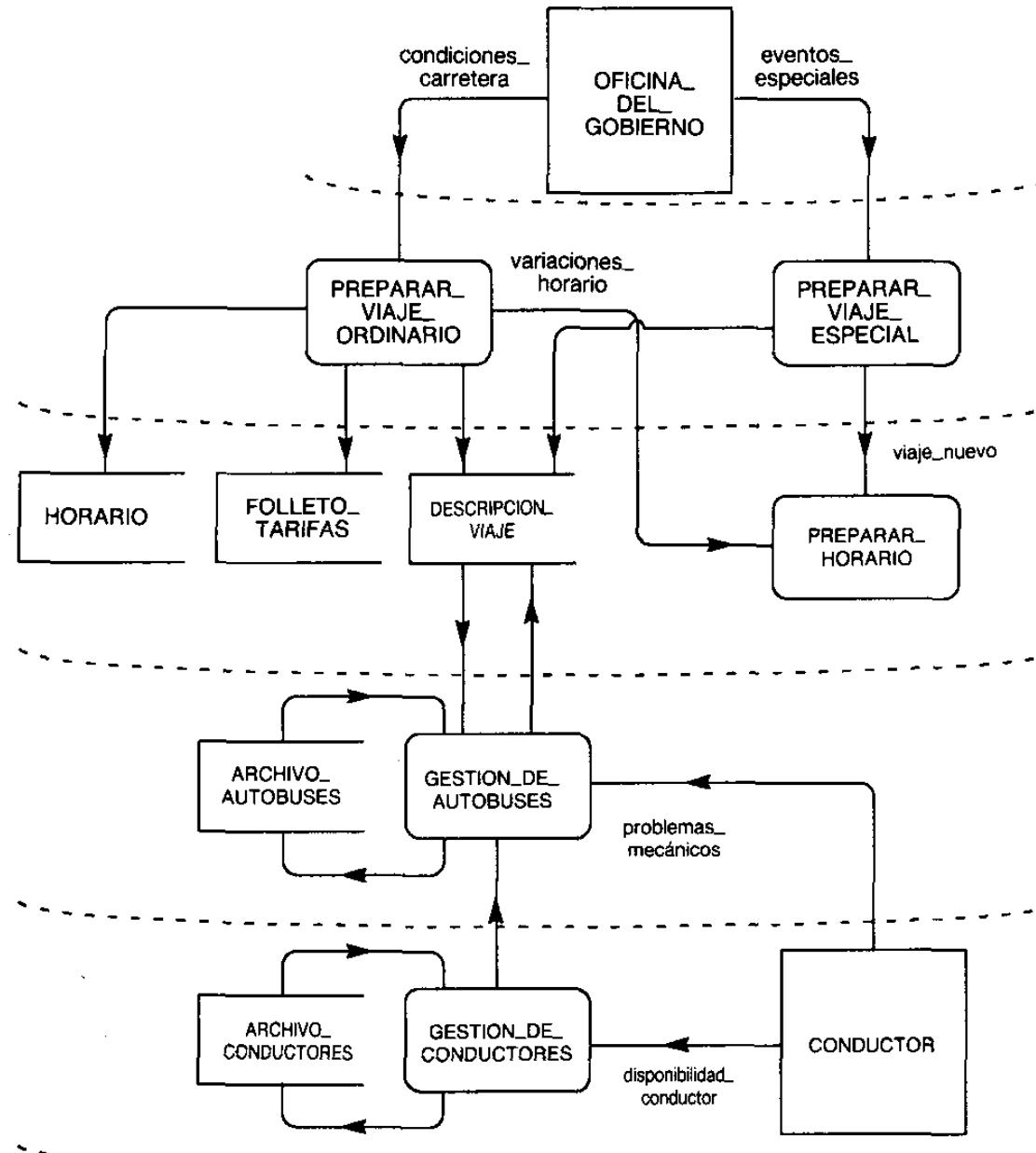


(a) Esquema F para GESTION_DE_PASAJEROS

Figura 10.4. Refinamientos del esquema F: primer refinamiento.

Ahora se considera GESTION_DE_SERVICIOS. Otra vez, se usa una estrategia centrífuga, comenzando por las interfaces. La interfaz OFICINA_DEL_GOBIERNO comunica las condiciones de las carreteras (que afectan los viajes ordinarios) y los eventos especiales (que pueden causar la creación de viajes especiales). Así, se determinan dos procesos separados, denominados PREPARAR_VIAJE_ORDINARIO y PREPARAR_VIAJE_ESPECIAL. El primero se encarga de crear el HORARIO, el FOLLETO_TARIFAS y la DESCRIPCION_VIAJE; este último almacén de datos es también afectado por el proceso PREPARAR_VIAJE_ESPECIAL cuando define viajes especiales. Por último, tanto PREPARAR_VIAJE_ORDINARIO como PREPARAR_VIAJE_ESPECIAL se comunican con el proceso PREPARAR_HORARIO; el primero comunica las variaciones de los horarios de los viajes ordinarios, el último la creación de viajes especiales.

Por otro lado, si nos concentráramos en la interfaz CONDUCTOR, se observa que intercambia dos flujos de información con la oficina de gestión de servicios. Uno, DISPONIBILIDAD_CONDUCTOR, se dirige a un proceso cuyo propósito principal es controlar a los conductores, denominado GESTION_DE_CONDUCTORES. El otro, PROBLEMAS_MECANICOS, se dirige a un proceso denominado GESTION_DE_AUTOBUSES, que se encarga de controlar los autobuses y también



(b) Esquema F para GESTION_DE_SERVICIOS

Figura 10.4.Bis Refinamientos del esquema F: primer refinamiento (*continuación*).

de preparar viajes completando **DESCRIPCION_VIAJE** con la selección del conductor y del autobús. Esto es posible porque el proceso tiene información disponible sobre los autobuses (el almacén de datos **ARCHIVO_AUTOBUSES**) y recibe

del proceso GESTION_DE_CONDUCTORES información sobre la disponibilidad de conductores. El esquema F refinado se muestra en la figura 10.4b.

Ahora se producirán los esquemas externos para los cinco almacenes de datos: DESCRIPCION_VIAJE, FOLLETO_TARIFAS, HORARIO, ARCHIVO_AUTOBUSES, ARCHIVO_CONDUCTORES. Están ilustrados en la figura 10.5. Obsérvese que DESCRIPCION_VIAJE se refiere a un VIAJE_DIARIO específico, que está compuesto de varios SEGMENTO_DE_RECORRIDO_DIARIO. Considérese también que se puede reservar cada segmento independientemente, pero por ahora no se considerarán las reservas en el esquema externo de DESCRIPCION_VIAJE. En cambio, los esquemas externos HORARIO y FOLLETO_TARIFAS consideran un viaje genérico, uno que se repite varios días a la semana, e indican el tiempo y el costo de cada segmento genérico. Obviamente, la interrelación COMPUESTO_POR entre VIAJE y SEGMENTO_DE_RECORRIDO es de uno a muchos, tanto para un viaje diario como para un viaje genérico. Los esquemas externos de ARCHIVO_AUTOBUSES y ARCHIVO_CONDUCTORES indican que cada autobús puede tener varios PROBLEMA_AUTOBUS y que cada CONDUCTOR puede tener varios días de AUSENCIA_CONDUCTOR.

10.3.2. Esquema D

Considérese la entidad VIAJE: los esquemas externos en las figuras 10.5a y 10.5b indican dos direcciones de refinamiento para esta entidad. Primero, se observa que cada cliente tiene una reserva o viaja en un VIAJE_DIARIO específico, en tanto que el concepto de VIAJE es más general; las propiedades de VIAJE abarcan el recorrido y el horario, mientras que las de VIAJE_DIARIO incluyen su fecha, el conductor, y el autobús. Se supone que cada viaje opera una vez al día, no más. En segundo lugar, se observa que cada cliente tiene una reserva o viaja por uno o más segmentos del recorrido del viaje. Las propiedades de un segmento del recorrido incluyen la ciudad de salida y la de llegada, la distancia y el precio. Cada viaje tiene un número variable de segmentos, uno por cada parada intermedia. Los pasajeros pueden tener reservas y viajar en múltiples segmentos del mismo viaje.

Obsérvese que estas dos direcciones de refinamiento han sido presentadas de algún modo en los esquemas externos de la figura 10.5. En efecto, la figura 10.6f muestra la integración del esquema externo de la figura 10.5a con los esquemas externos de la figura 10.6b o 10.6c. La interrelación DE provee la integración. De este modo se llega al final con cuatro entidades: VIAJE, VIAJE_DIARIO, SEGMENTO_DE_RECORRIDO, y SEGMENTO_DE_RECORRIDO_DIARIO. Se requieren todas las entidades, porque cada una corresponde a un concepto diferente que tiene sus propias características. Los dos casos de la interrelación COMPUESTO_POR tienen cardinalidades (1, 1) y (1, m) ya que cada segmento pertenece exactamente a un viaje, mientras que cada viaje tiene uno o más segmentos. De forma similar, los dos casos de la interrelación DE tienen cardinalidades (1, 1) y (1, m) ya que cada viaje diario corresponde exactamente a un viaje, mientras que cada

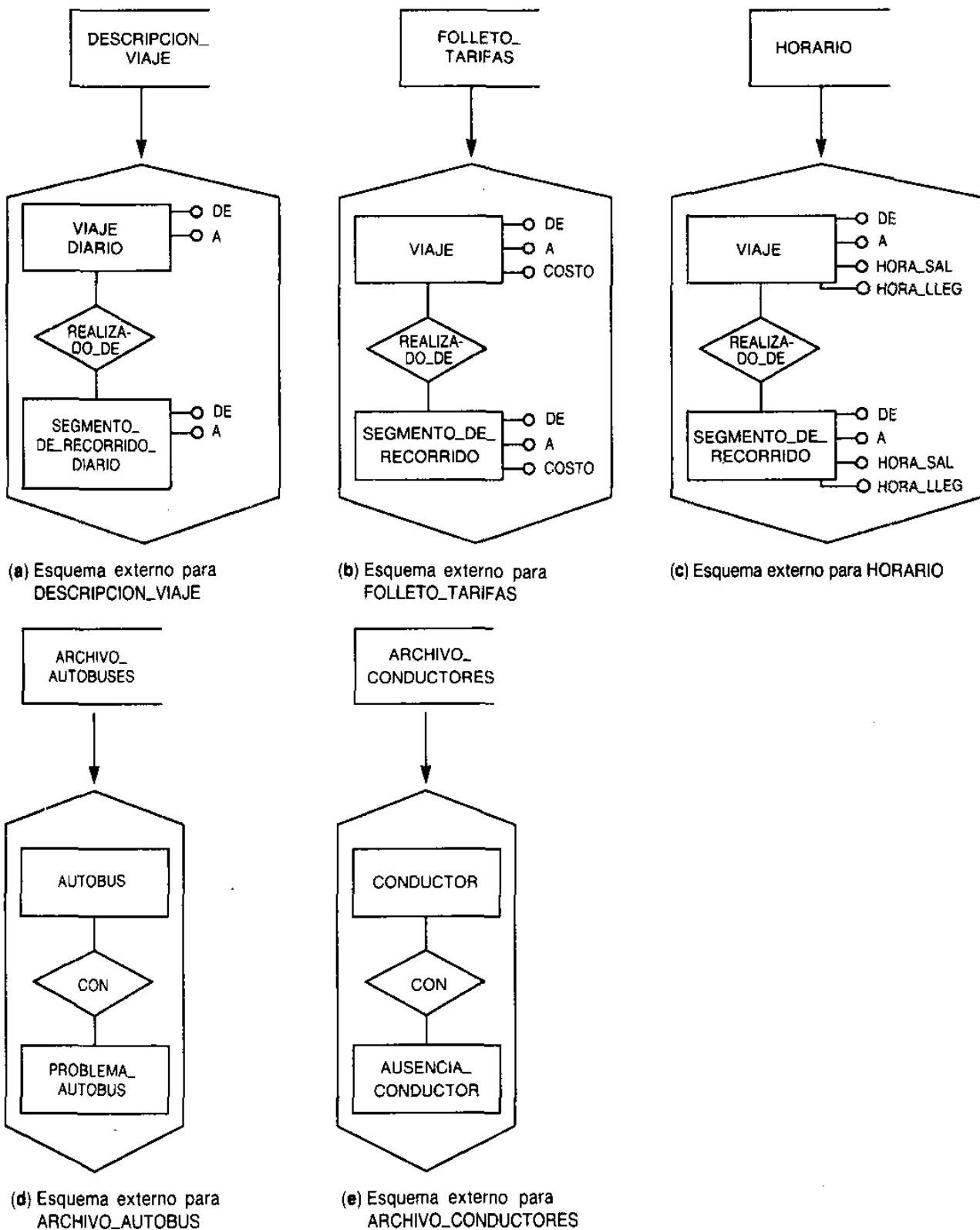
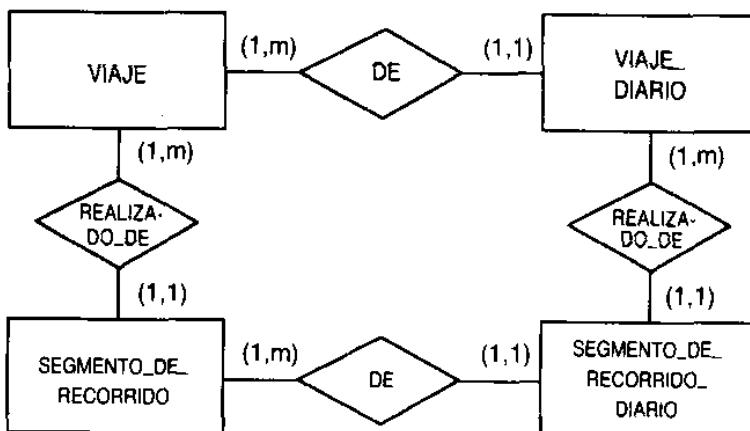
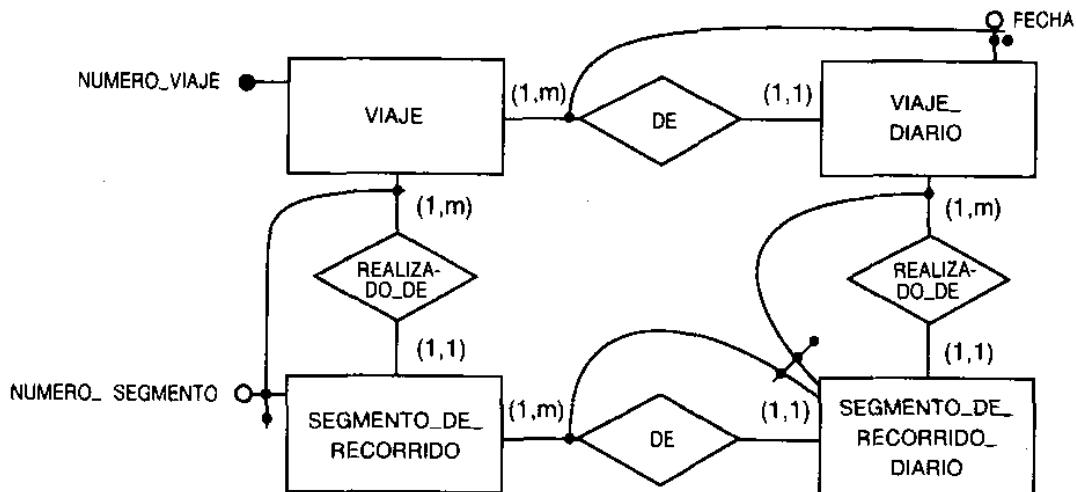


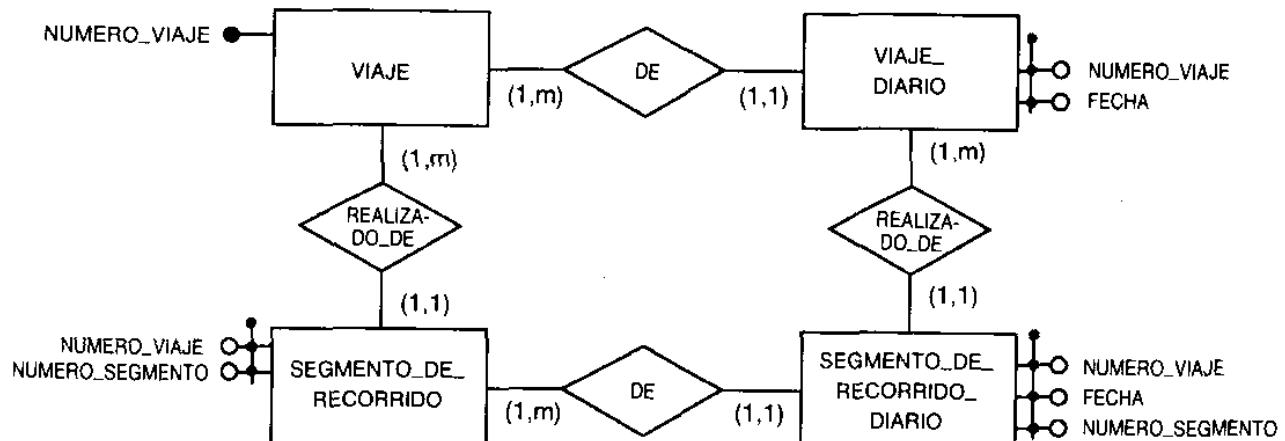
Figura 10.5. Esquemas externos para los almacenes de datos del esquema F de la figura 10.4.



(a) Integración de los esquemas externos de DESCRIPCION_VIAJE y FOLLETO_TARIFAS

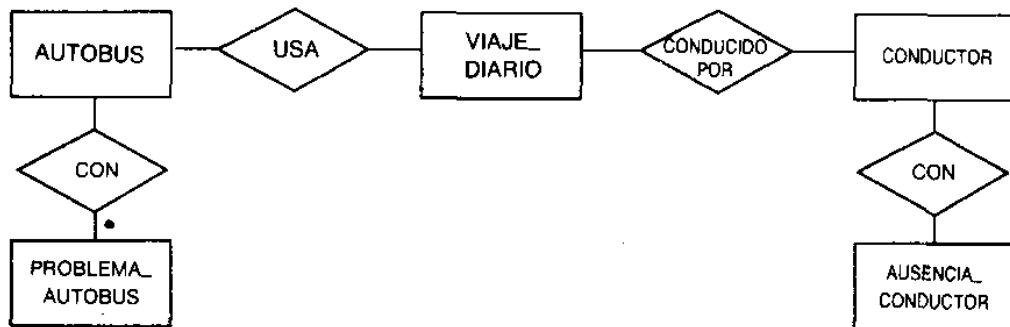


(b) Subesquema que describe los viajes y segmentos de recorrido con identificadores externos

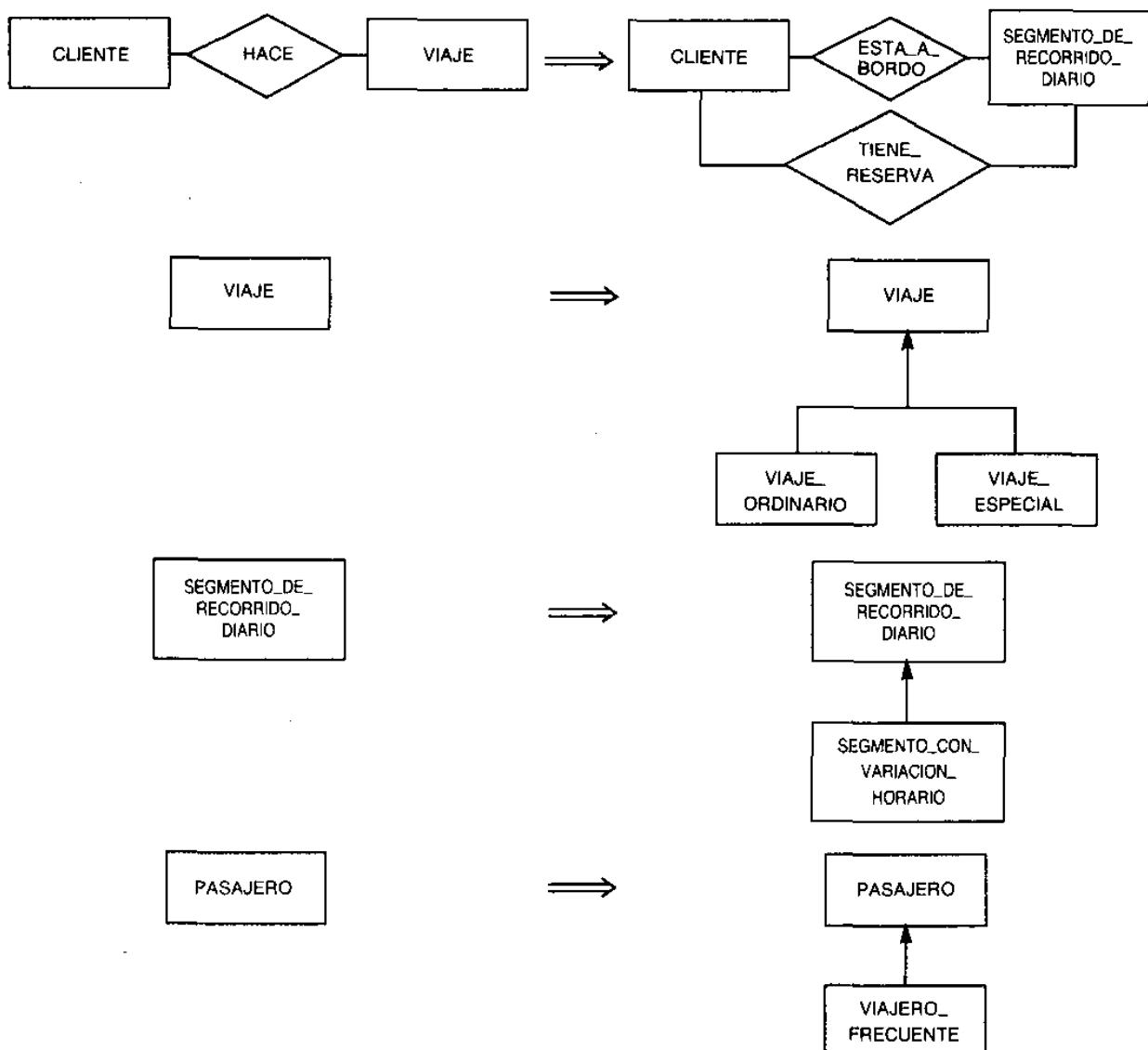


(c) Subesquema que describe viajes y segmentos de recorrido con identificadores internos

Figura 10.6. Refinamientos del esquema D: primer refinamiento.

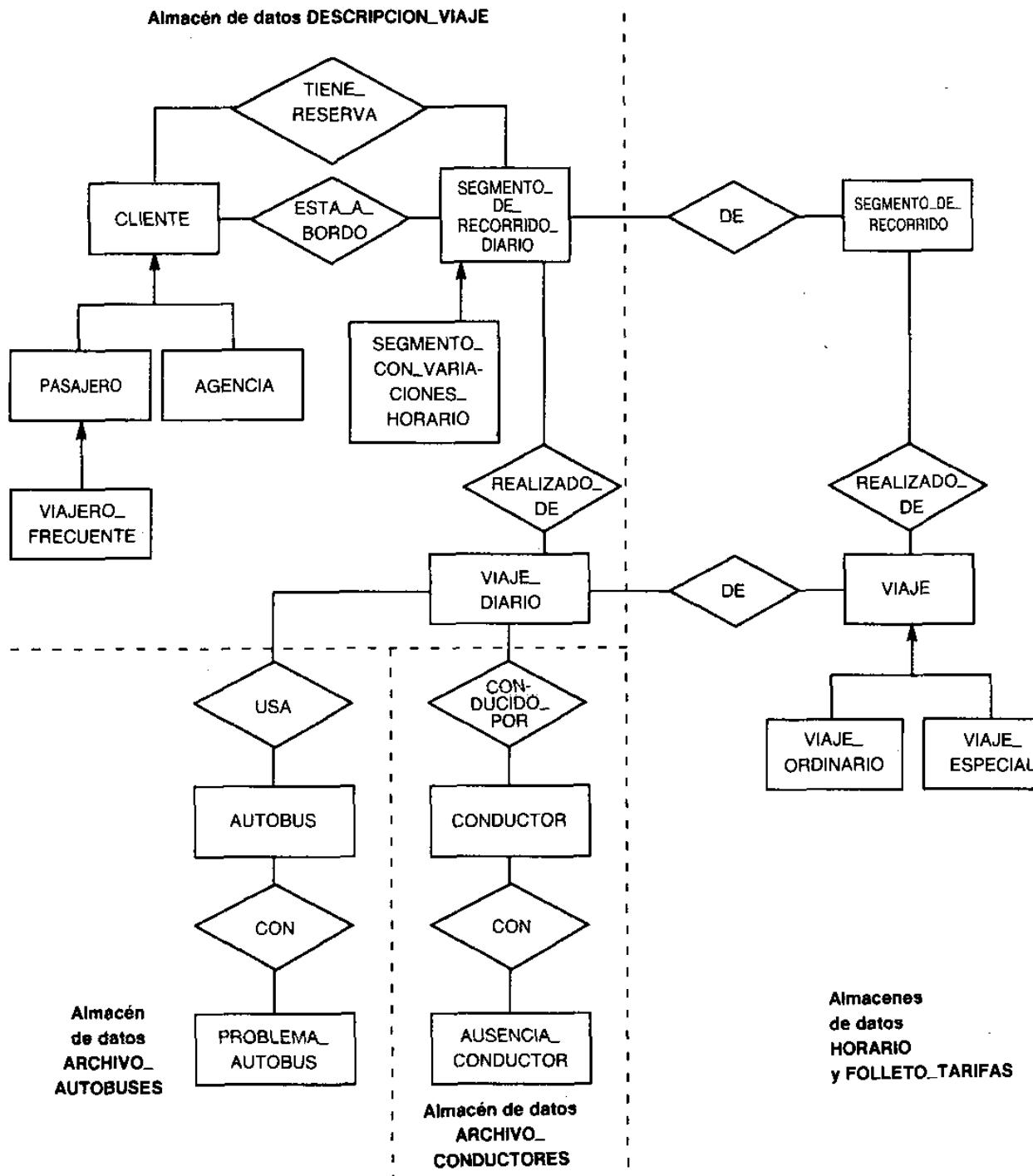


(d) Integración de los esquemas externos de ARCHIVO_AUTOBUSES y ARCHIVO_CONDUCTORES con la entidad VIAJE_DIARIO



(e) Refinamientos descendentes

Figura 10.6.Bis Refinamientos del esquema D: primer refinamiento (*continuación*).



(f) Esquema D integrado con los esquemas externos de almacenes de datos superpuestos

Figura 10.6.2Bis Refinamientos del esquema D: primer refinamiento (continuación).

viale corresponde a multiples viajes diarios. Las cuatro entidades se representan en la figura 10.6a.

Ahora comentamos la identificación de estas entidades, para clarificar un poco más su significado. VIAJE se identifica por NUMERO_VIAJE; VIAJE_DIARIO se identifica externamente por FECHA y por la entidad VIAJE a través de la interrelación DE; SEGMENTO_DE_RECORRIDO por NUMERO_SEGMENTO y por la entidad VIAJE a través de la interrelación COMPUESTO_>; y SEGMENTO_DE_RECORRIDO_DIARIO se identifica externamente por la entidad VIAJE_DIARIO a través de la interrelación COMPUESTO_POR y por la entidad SEGMENTO_DE_RECORRIDO a través de la interrelación DE. Esta situación se representa en la figura 10.6b. Dado que esa identificación externa no es obvia para muchos lectores, también se suministran identificadores internos para cada entidad, como se muestra en la figura 10.6c. Así, VIAJE se identifica por NUMERO_VIAJE; VIAJE_DIARIO, por el par (NUMERO_VIAJE, FECHA); SEGMENTO_DE_RECORRIDO_DIARIO, por el par (NUMERO_VIAJE, NUMERO_SEGMENTO) y SEGMENTO_DE_RECORRIDO_DIARIO, por el trío (NUMERO_VIAJE, NUMERO_SEGMENTO, FECHA).

Consideremos los dos esquemas externos que quedan (Figs. 10.5d y 10.5e). Se integran al usar VIAJE_DIARIO como la entidad (escogida entre las cuatro anteriores mencionadas) a la cual se debe conectar la información de AUTOBUS y CONDUCTOR. La integración se muestra en la figura 10.6d.

En seguida, se analizan los requerimientos para descubrir aspectos que todavía no aparezcan en el esquema D actual, y así se producen varios refinamientos descendentes (Fig. 10.6.e). Considérese la interrelación HACE: se reconoce que un cliente y un viaje pueden estar conectados de dos maneras: 1) un cliente tiene una reserva (TIENE_RESERVA) para un cierto viaje; y 2) un cliente ESTA_A_BORDO de cierto viaje. En este caso, viaje quiere decir realmente SEGMENTO_DE_RECORRIDO_DIARIO, que es la unidad más pequeña que se puede reservar o viajar. Se nota entonces cómo VIAJE puede dividirse en VIAJE_ORDINARIO y VIAJE_ESPECIAL, y se introduce una jerarquía de generalización (Fig. 10.6e). Se descubre también que el flujo de información acerca de condiciones-carretera en el esquema F puede afectar el horario de segmentos de viajes en fechas específicas; por tanto, se introduce la entidad (diaria) SEGMENTO_CON_VARIACION_HORARIO como un subconjunto de SEGMENTO_DE_RECORRIDO_DIARIO, de manera que se pueda representar variaciones en el horario (Fig. 10.6e). Finalmente, se considera PASAJERO y se reconoce que algunos de ellos están inscritos en el programa VIAJERO_FRECUENTE, lo que causa la introducción de una nueva entidad subconjunto.

La figura 10.6f muestra el esquema D refinado producto de todos estos cambios. Obsérvese que el esquema está dividido en cuatro sectores; cada uno de ellos corresponde a un almacén de datos y, en efecto, todos los conceptos de los esquemas externos de los almacenes de datos están presentes en el esquema D. Esta verificación de compleción, aunque realizada en una etapa intermedia, es muy útil para asegurar al diseñador que los esquemas D y F son coherentes y están completos en niveles comparables de abstracción.

10.4. Segundo refinamiento

El segundo refinamiento se realiza también primero para el esquema F y luego para el esquema D; en los dos casos este refinamiento produce el esquema final.

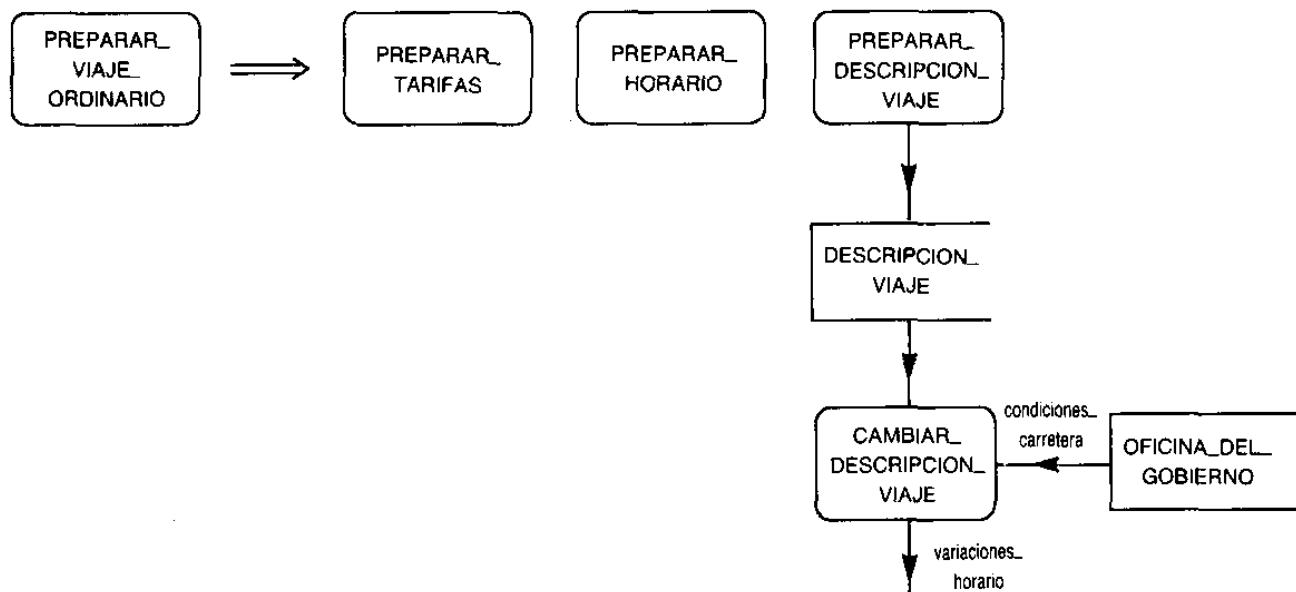
10.4.1. Esquema F

En primer lugar, se refinan algunos procesos de los esquemas F de GESTION_DE_SERVICIOS (DFD de la figura 10.4b); luego se integran los dos esquemas F producidos hasta ahora. Considérese el proceso PREPARAR_VIAJE_ORDINARIO. Se distinguen tres grupos independientes de actividades: PREPARAR_TARIFAS, PREPARAR_HORARIO y PREPARAR_DESCRIPCION_VIAJE. También se observa que es necesario el proceso CAMBIAR_DESCRIPCION_VIAJE para producir nuevos horarios cuando las condiciones de las carreteras son malas. De igual forma, se distinguen dos actividades independientes para el proceso PREPARAR_VIAJE_ESPECIAL: PREPARAR_DESCRIPCION_VIAJE y ANUNCIAR_VIAJE_NUEVO. Estos refinamientos se muestran en las figuras 10.7a y 10.7b.

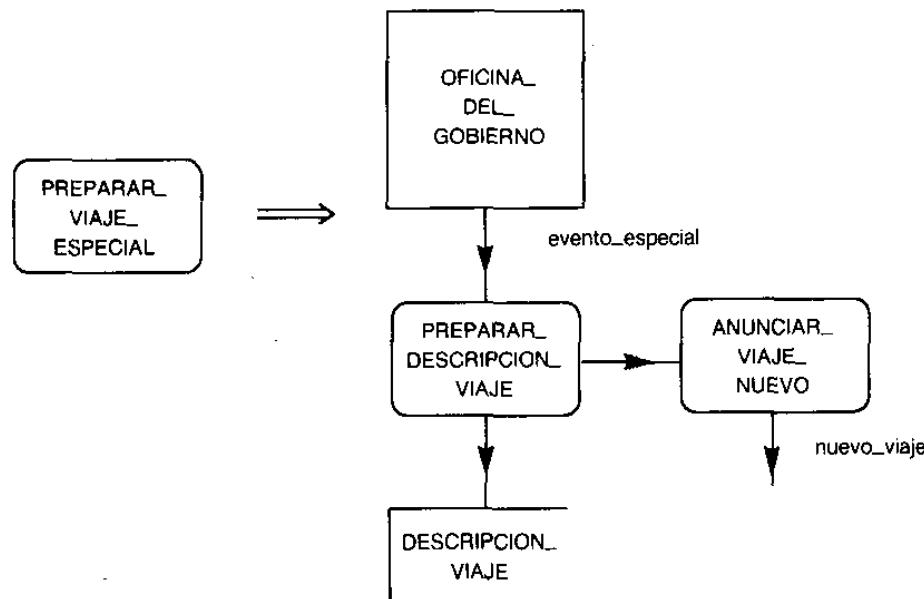
Por último se considera el proceso GESTION_DE_AUTOBUSES. En él, se distinguen claramente dos actividades: GESTION_DE_VIAJES_DIARIOS y GESTION_DE_AUTOBUSES. La primera se encarga de la organización de un viaje específico al asociarlo con un autobús y conductor; la segunda se encarga de gestionar la compra, reparación, y venta de los autobuses. Obsérvese que se genera una conexión entre DESCRIPCION_VIAJE y CONDUCTORES, que corresponde a la necesidad de darles información completa sobre el viaje; este refinamiento es ascendente.

Para completar el esquema F se requiere la integración de los dos esquemas F de las figuras 10.4a y 10.4b (modificados después de los refinamientos anteriores), guiada por el esquema F armazón. La conexión entre la oficina de GESTION_DE_PASAJEROS y la de GESTION_DE_SERVICIOS la dan los almacenes de datos FOLLETO_TARIFAS, HORARIO, y DESCRIPCION_VIAJE, utilizadas por la primera y producidas por la segunda. La mayor parte de la información intercambiada entre las dos oficinas es permanente, y por ello se representa a través de almacenes de datos; aun así, la oficina de GESTION_DE_SERVICIOS comunica las variaciones de horarios y los viajes nuevos a la actividad PREPARAR_HORARIO, que a su vez actualiza los horarios de la compañía y notifica las variaciones a los clientes. El esquema F final que se produce por integración se muestra en la figura 10.8.

Cada DESCRIPCION_VIAJE, de hecho, es producida por PREPARAR_DESCRIPCION_VIAJE_ORDINARIO o bien por PREPARAR_DESCRIPCION_VIAJE_ESPECIAL, y la usa HACER_RESERVAS a fin de añadir reservas para el viaje correspondiente, también la usa GESTION_DE_AUTOBUSES para añadir la información final sobre los viajes y entregársela al conductor. Por ello, este documento es de suma importancia; de hecho, es el núcleo de la base de datos que se introducirá como resultado del diseño.



(a) Refinamiento descendente de PREPARAR_VIAJE_ORDINARIO



(b) Refinamiento descendente de PREPARAR_VIAJE_ESPECIAL

Figura 10.7. Refinamientos del esquema F: segundo refinamiento.

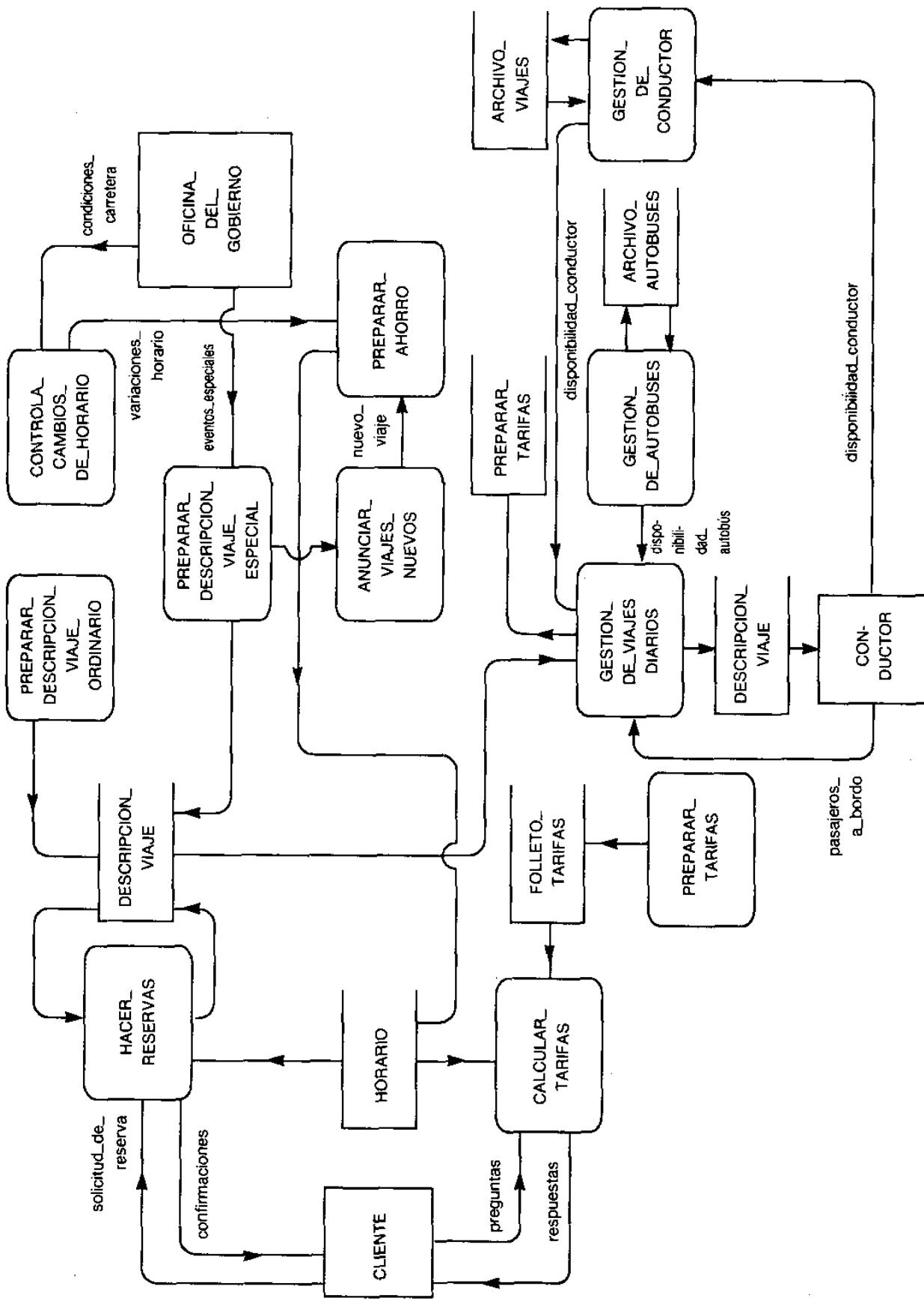


Figura 10.8. Esquema F final.

10.4.2. Esquema D

El esquema D de la figura 10.6f no necesita entidades, interrelaciones o subconjuntos adicionales; el esquema D se completa especificando los atributos, cardinalidades mínimas y máximas, e identificadores. Los refinamientos progresivos pueden entenderse observando la figura 10.9 directamente.

Se comienza con la entidad CLIENTE y se considera el par NOMBRE y TELEFONO como identificador de CLIENTE. Adicionalmente, VIAJERO_FRECUENTE tiene el atributo kilometraje, que da el número de kilómetros acumulados hasta el momento (porque los pasajeros reciben un premio después de un cierto número de kilómetros). Las dos interrelaciones TIENE_RESERVA y ESTÁ_A_BORDO tienen las mismas cardinalidades (0, m), ya que cada cliente puede tener cero o más reservas o estar a bordo para múltiples segmentos diarios de viaje y cada segmento diario de un viaje puede ser reservado y tener a bordo cero o más clientes. TIENE_RESERVA posee dos atributos: OPCION_FUMAR y NUM_ASIENTO.

Considérese ahora el agrupamiento de entidades VIAJE, VIAJE_DIARIO, SEGMENTO_RECORRIDO, SEGMENTO_DE_RECORRIDO_DIARIO.

1. VIAJE se identifica por el atributo NUMERO_VIAJE. El atributo DIAS_DE_SEMANA indica los días de la semana (digamos, como una serie de dígitos o caracteres) en los cuales el viaje se realiza. Otros atributos son CIUDAD_SA-LIDA, CIUDAD_LLEGADA y el NUMERO_DE_SEGMENTOS en que se descomponen el viaje.
2. VIAJE_DIARIO se identifica por NUMERO_VIAJE y FECHA. Esta entidad no tiene otros atributos; sin embargo, está conectada a través de la interrelación USA a AUTOBUS, y a través de la interrelación CONDUCIDO_POR a la entidad CONDUCTOR. Se supone que se asigna un autobús y un conductor a un viaje completo.
3. SEGMENTO_DE_RECORRIDO se identifica por NUMERO_VIAJE y NUMERO_SEGMENTO, un número progresivo dado a los segmentos del viaje. Los atributos de SEGMENTO_DE_RECORRIDO incluyen CIUDAD_SALIDA y HORA_SALIDA, CIUDAD_LLEGADA y HORA_LLEGADA, PRECIO y DISTANCIA.
4. SEGMENTO_DE_RECORRIDO_DIARIO se identifica por NUMERO_VIAJE, NUMERO_SEGMENTO y FECHA. Tiene otros dos atributos, ASIENTOS_DISPONIBLES y ASIENTOS_RESERVEDOS.

Examinemos ahora el agrupamiento de las entidades AUTOBUS y PROBLEMA_AUTOBUS y las interrelaciones CON y USA. La interrelación USA tiene cardinalidades (0, m) y (1, 1), ya que cada AUTOBUS se asigna dinámicamente a un viaje diario, pero existe la posibilidad de que no esté asignado a ninguno (por ejemplo, cuando está en reparación); en cambio, cada viaje diario requiere exactamente un AUTOBUS. La entidad AUTOBUS se identifica por su ID_AUTOBUS o bien por su NUMERO_MATRICULA; tiene los atributos MARCA, ASIENTOS y ULTIMA_REVISION. La entidad PROBLEMA_AUTOBUS se identifica por el atributo NUMERO_PROBLEMA, y tiene el atributo DESCRIPCION. La interrelación CON es

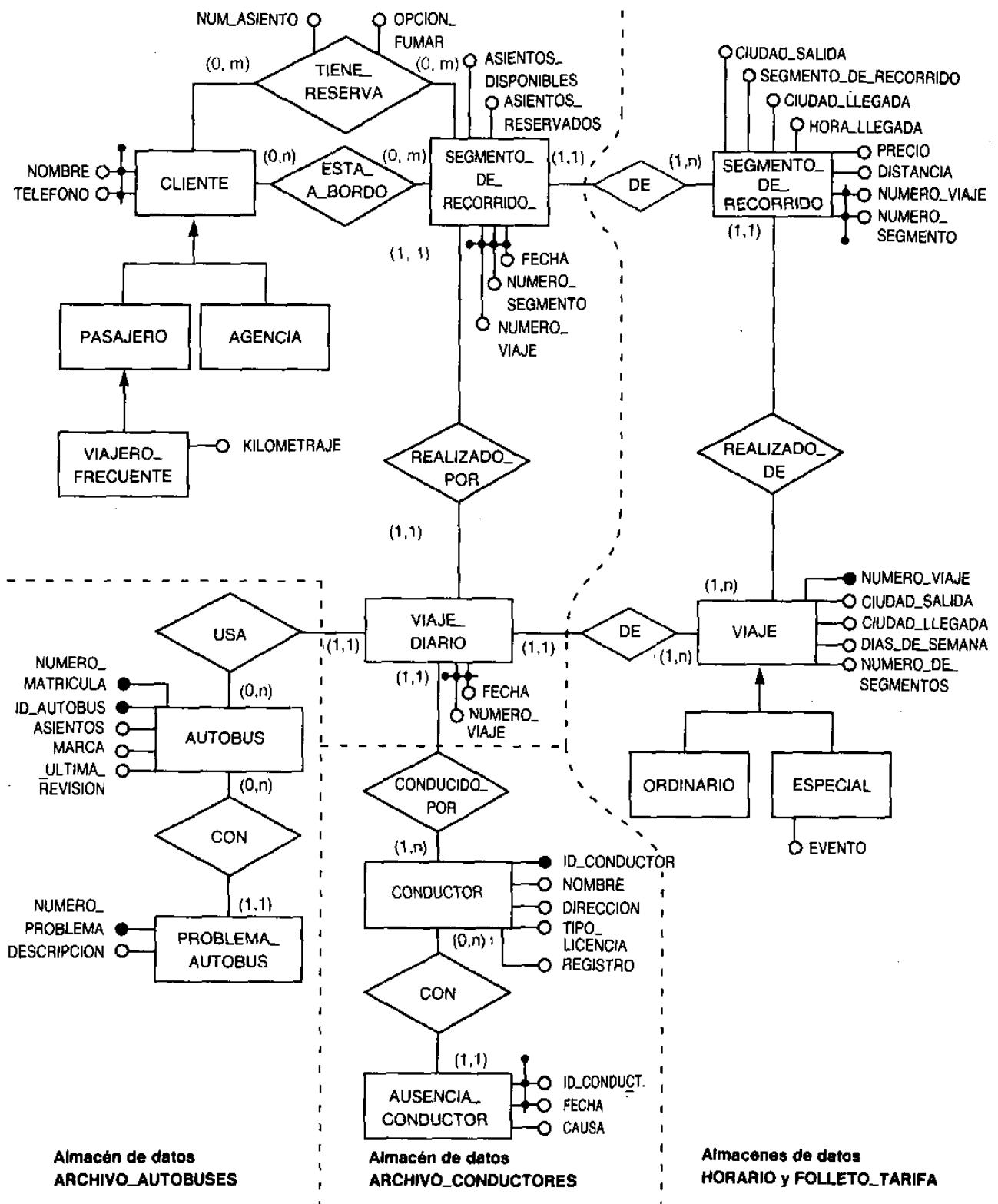
Almacén de datos DESCRIPCION_VIAJE**Figura 10.9.** Esquema D final.

Tabla 10.1. Reglas de cálculo para los datos derivados en el esquema D final (Fig. 10.9)

Regla núm.	Contenido
Regla 1	ASIENTOS_DISPONIBLES de SEGMENTO_DE_RECORRIDO = ASIENTOS del AUTOBUS que USA el VIAJE_DIARIO de ese SEGMENTO_DE_RECORRIDO
Regla 2	ASIENTOS_RESERVADOS de SEGMENTO_DE_RECORRIDO_DIARIO = cardinalidad de CLIENTE que TIENE_RESERVA en ese SEGMENTO_DE_RECORRIDO_DIARIO
Regla 3	NUMERO_DE_SEGMENTOS de VIAJE = cardinalidad de SEGMENTO_DE_RECORRIDO_REALIZADO_DE ese VIAJE
Regla 4	CIUDAD_SALIDA de VIAJE = CIUDAD_SALIDA de SEGMENTO_DE_RECORRIDO con NUMERO_SEGMENTO = 1 de ese VIAJE
Regla 5	CIUDAD_LLEGADA de VIAJE = CIUDAD_LLEGADA de SEGMENTO_DE_RECORRIDO con más alto NUMERO_de ese VIAJE
Regla 6	CIUDAD_SALIDA de SEGMENTO_DE_RECORRIDO con (NUMERO_SEGMENTO = i, donde i > 1) y (NUMERO_VIAJE=j)=CIUDAD_LLEGADA de SEGMENTO_DE_RECORRIDO con (NUMERO_SEGMENTO=i - 1) y (NUMERO_VIAJE=j)

(0, m), (1, 1), ya que cada autobús puede tener cero o más problemas, mientras que cada problema se refiere exactamente a un autobús.

Finalmente se considera el grupo de las entidades CONDUCTOR y AUSENCIA_CONDUCTOR y las interrelaciones CONDUCIDO_POR y CON. No se analizan las cardinalidades de las interrelaciones, pero son similares a las de USA y CON, que ya examinamos. Cada CONDUCTOR se identifica por una ID_CONDUCTOR y tiene NOMBRE, DIRECCION, TIPO_LICENCIA y un REGISTRO (PERMISO) de conducir. Cada AUSENCIA_CONDUCTOR está identificada por el par (ID_CONDUCTOR y FECHA), y tiene el atributo CAUSA.

El esquema D final se muestra en la figura 10.9. Obsérvese que varios atributos son derivados, porque sus valores pueden ser calculados usando los valores de los atributos conectados. Las reglas de cálculo para los datos derivados se muestran en la tabla 10.1.

10.5. Verificación de compleción

El caso de estudio se termina mostrando que los esquemas F y D finales son mutuamente completos. La compleción del esquema D se comprueba mediante la verificación de que cada concepto que se expresa en los flujos de datos y almacenes de datos del esquema F aparece en el esquema D. La compleción del esquema F se comprueba verificando que todas las operaciones que deben estar expresadas para los datos del esquema D aparecen también como procesos del esquema F.

10.5.1. Compleción del esquema D

Los flujos de datos del esquema F de la figura 10.8 se usan para modificar el contenido de los almacenes de datos, y no llevan información adicional respecto a los mismos; en consecuencia, se puede realizar la comprobación teniendo en cuenta sólo los almacenes de datos y verificando que exista un grupo apropiado de entidades e interrelaciones del esquema D que represente el mismo contenido.

1. La DESCRIPCION_VIAJE se representa por el siguiente grupo de entidades e interrelaciones: CLIENTE, TIENE_RESERVA, ESTA_A_BORDO, SEGMENTO_DE_RECORRIDO_DIARIO, REALIZADO_DE, VIAJE_DIARIO, USA, AUTOBUS, CONDUCIDO_POR y CONDUCTOR.
2. FOLLETO_TARIFAS y HORARIO se representan por el siguiente grupo de entidades e interrelaciones: SEGMENTO_DE_RECORRIDO, REALIZADO_DE y VIAJE. En concreto, el atributo PRECIO da la tarifa para cada segmento de un viaje, y los atributos HORA_SALIDA y HORA_LLEGADA dan la información sobre el horario.
3. El ARCHIVO_AUTOBUSES se representa por el siguiente grupo de entidades e interrelaciones: AUTOBUS, CON y PROBLEMA_AUTOBUS.
4. El ARCHIVO_CONDUCTORES se representa por el siguiente grupo de entidades e interrelaciones: CONDUCTOR, CON, AUSENCIA_CONDUCTOR.

Las correspondencias entre los almacenes de datos y las porciones del esquema conceptual se representan en la figura 10.9.

10.5.2. Compleción del esquema F

Para cada entidad e interrelación del esquema D, se necesita verificar que exista, al menos, un proceso encargado de su creación y uso.

1. El proceso HACER_RESERVAS crea, recupera y modifica las entidades CLIENTE y SEGMENTO_DE_RECORRIDO_DIARIO y las interrelaciones TIENE_RESERVA y ESTA_A_BORDO.
2. Los procesos PREPARAR_DESCRIPCION_VIAJE_ORDINARIO y PREPARAR_DESCRIPCION_VIAJE_ESPECIAL crean las entidades VIAJE_DIARIO y SEGMENTO_DE_RECORRIDO_DIARIO y las interrelaciones DE y REALIZADO_DE. Estas entidades e interrelaciones son usadas por los procesos HACER_RESERVAS y GESTION_DE_VIAJES_DIARIOS.
3. Los procesos PREPARAR_TARIFAS y PREPARAR_HORARIO crean las entidades VIAJE y VIAJE_DIARIO con la interrelación DE entre ellas, mismas que son usadas por los procesos HACER_RESERVAS y PREPARAR_TARIFAS.
4. El proceso GESTION_DE_AUTOBUSES crea y usa las entidades AUTOBUS y PROBLEMA_AUTOBUS y las interrelaciones USA y CON.

5. El proceso GESTION_DE_CONDUCTORES crea y usa las entidades CONDUCTOR y AUSENCIA_CONDUCTOR y la interrelación CON; el proceso GESTION_DE_AU-TOBUSES crea y usa la interrelación CONDUCIDO_POR.

10.6. Esquemas de navegación

En este apartado sólo se considerará el proceso HACER_RESERVAS; el análisis de todos los otros procesos y la especificación de todas las operaciones en el caso de estudio se deja a los lectores como ejercicio. Se identifican varias operaciones que ejecuta el proceso HACER_RESERVAS:

1. BUSCAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA.
2. BUSCAR SEGMENTO_DE_RECORRIDO POR AMBAS_CIUDAD_SALIDA_Y_CIUDAD_LLEGADA.
3. BUSCAR TODOS LOS VIAJES QUE TENGAN UNA PARADA INTERMEDIA EN UNA CIUDAD DADA.
4. CREAR UN REGISTRO DE CLIENTE NUEVO.
5. HACER UNA RESERVA (para un cliente que ya existe).
6. BORRAR RESERVAS DE UN VIAJE PASADO.
7. BORRAR EL REGISTRO DE UN CLIENTE (porque no es un viajero frecuente o no tiene reservas).
8. CALIFICAR A UN CLIENTE COMO VIAJERO FRECUENTE.
9. RECUPERAR EL NUMERO DE KILOMETROS ACUMULADOS POR VIAJEROS FRECUENTES.
10. ACTUALIZAR EL KILOMETRAJE DE UN VIAJERO FRECUENTE.
11. RECUPERAR TODAS LAS RESERVAS ACTUALES PARA UN VIAJE DETERMINADO EN UNA FECHA DETERMINADA.
12. RECUPERAR TODAS LAS RESERVAS ACTUALES DE UNA AGENCIA DETERMINADA.

Las primeras tres operaciones arriba citadas son consultas al HORARIO; el encargado de reservas las puede combinar para generar consultas que puedan resolver el problema general de encontrar el viaje que conecte mejor dos ciudades para un cliente dado (obsérvese que los esquemas de navegación para O₂ y O₃ son idénticos). Las operaciones 4-7 proveen el control de reservas; las operaciones 8-10 proveen el control de información de los viajeros frecuentes; las operaciones 11-12 preparan los informes que usan antes del viaje las agencias. Estas operaciones se describen en la siguiente lista, y los esquemas de navegación correspondientes se representan en las figuras 10.10 a 10.20. Los esquemas de operación se omiten por brevedad.

1. BUSCAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA.
Recupera la información que concierne a todos los segmentos de reco-

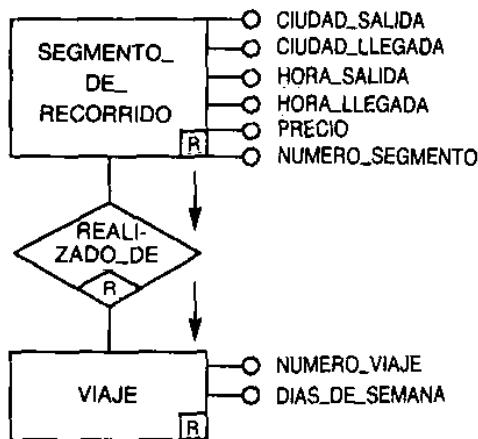


Figura 10.10. Esquema de navegación para O₁.

rrido que salen de una ciudad dada. Determina el viaje y segmento de recorrido; recupera la hora de salida, ciudad de llegada, hora de llegada y el precio. También recupera la información concerniente a los días de la semana en los cuales operan los viajes identificados.

2. **BUSCAR SEGMENTO_DE_RECORRIDO POR AMBAS, CIUDAD_SALIDA Y CIUDAD_LLEGADA.** Recupera la información concerniente a todos los segmentos de recorrido que conectan a dos ciudades. Determina el viaje y el segmento de recorrido; recupera la hora

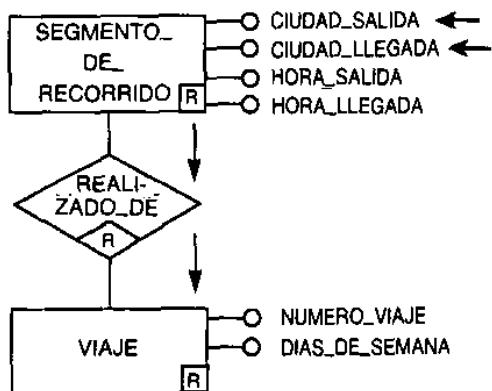


Figura 10.11. Esquema de navegación para O₂ y O₃.

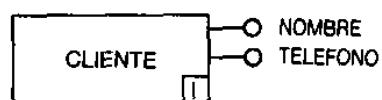


Figura 10.12. Esquema de navegación para O₄.

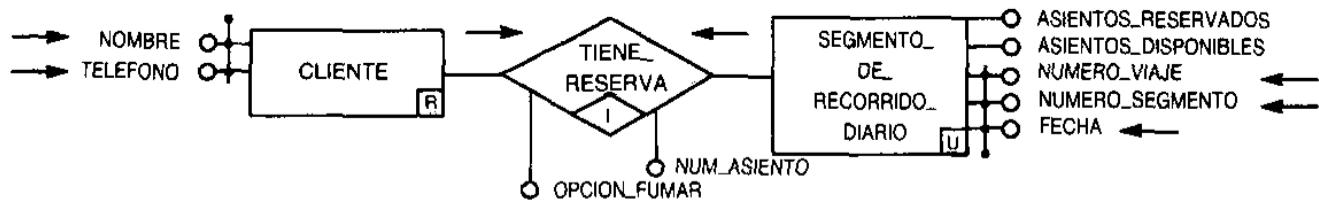


Figura 10.13. Esquema de navegación para O₅.

de salida, hora de llegada y el precio. También recupera la información concerniente a los días de la semana en los cuales operan los viajes identificados.

3. BUSCAR TODOS LOS VIAJES QUE TENGAN UNA PARADA INTERMEDIA EN UNA CIUDAD DADA. Determina la información concerniente a todos los viajes que tienen una ciudad dada como parada intermedia. Identifica el viaje y recupera las ciudades de salida y llegada, y las horas de salida y llegada del viaje. También recupera la información concerniente a los días de la semana en los cuales operan los viajes identificados.
4. CREAR UN REGISTRO DE CLIENTE NUEVO. Inserta un registro nuevo para un nuevo cliente, especificando NOMBRE y TELEFONO.



Figura 10.14. Esquema de navegación para O₆.

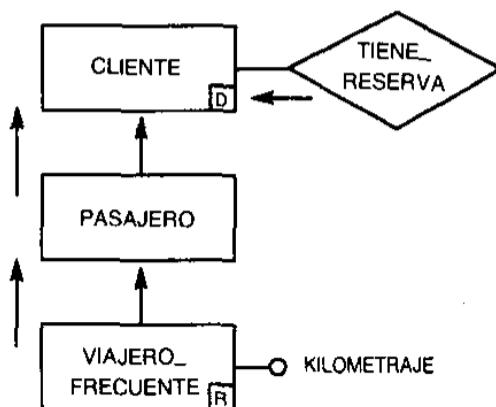


Figura 10.15. Esquema de navegación para O₇.

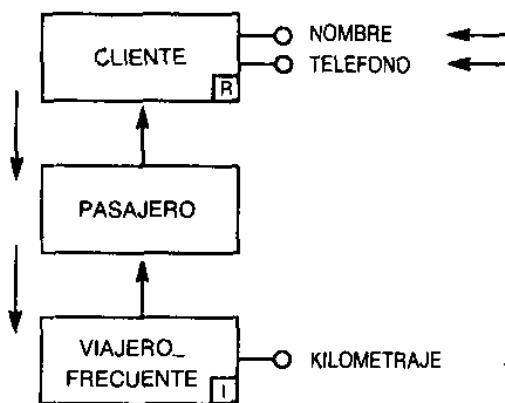


Figura 10.16. Esquema de navegación para O₈.

5. **HACER UNA RESERVA** (para un cliente existente). Obtiene acceso a la información relacionada con un segmento del recorrido, identificado por NUM_VIAJE, NUMERO_SEGMENTO y FECHA; verifica que haya asientos disponibles. Entonces, reduce los asientos disponibles, identifica al cliente e inserta la reserva, especificando el número de asientos reservados y la opción de fumar.
6. **BORRAR RESERVAS DE UN VIAJE PASADO**. Obtiene acceso a un SEGMENTO_DE_RECORRIDO_DIARIO específico de un viaje a través de su fecha y sus números de segmento y de viaje; después, obtiene acceso a toda la información de reservas conectada a él; por último, elimina todas las reservas.
7. **ELIMINAR EL REGISTRO DE UN CLIENTE** (porque no es un viajero frecuente, o no tiene reserva). Selecciona todos los clientes que no tienen reservas pendientes o que no son viajeros frecuentes y borra la información correspondiente.
8. **CALIFICAR A UN CLIENTE COMO VIAJERO FRECUENTE**. Ob-

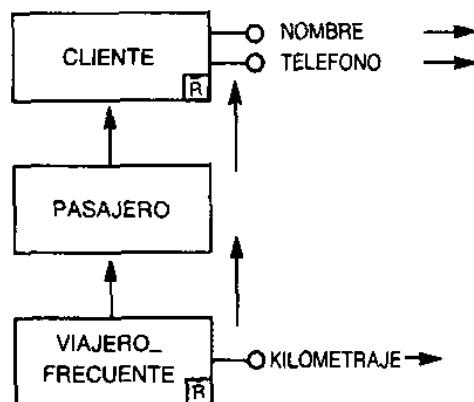


Figura 10.17. Esquema de navegación para O₉.

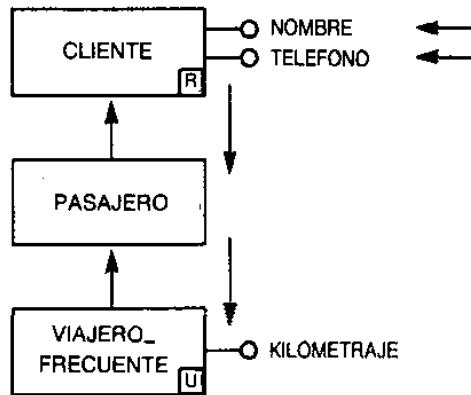


Figura 10.18. Esquema de navegación para O₁₀.

tiene acceso a un pasajero a través de NOMBRE y TELEFONO y modifica la situación del pasajero como viajero frecuente; fija el valor de kilometraje inicial en 500 kilómetros.

9. RECUPERAR LA CANTIDAD DE KILOMETROS ACUMULADA POR VIAJEROS FRECUENTES. Para todos los viajeros frecuentes, recupera el número de kilómetros acumulados y lo imprime en un informe.

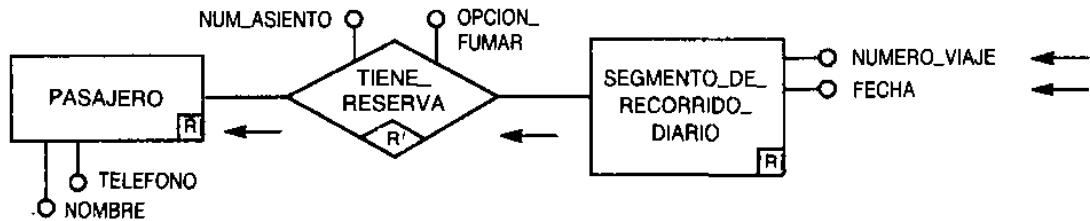


Figura 10.19. Esquema de navegación para O₁₁.

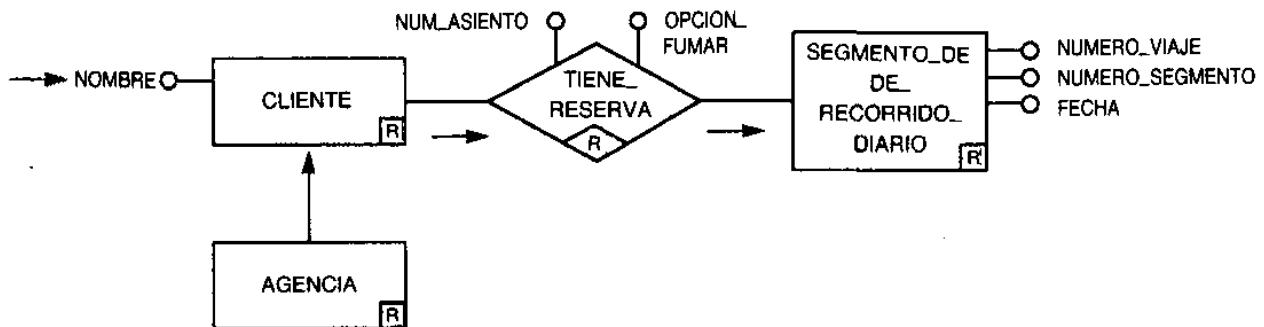


Figura 10.20. Esquema de navegación para O₁₂.

10. ACTUALIZAR EL KILOMETRAJE DE UN VIAJERO FRECUENTE. Para un pasajero dado, seleccionado a través de NOMBRE y TELEFONO, actualiza el kilometraje acumulado.
11. RECUPERAR TODAS LAS RESERVAS ACTUALES PARA UN VIAJE DETERMINADO EN UNA FECHA DETERMINADA. Para un cierto viaje en una fecha determinada, recupera los nombres y números de teléfono de los pasajeros o agencias que tengan reservas, con sus números de asiento y opciones de fumar.
12. RECUPERAR TODAS LAS RESERVAS ACTUALES DE UNA AGENCIA DETERMINADA. Para una cierta agencia, recupera todas las reservas indicando NUMERO_VIAJE, SEGMENTO_DE_RECORRIDO, y FECHA de las reservas, así como los números de asiento y opciones de fumar-no fumar escogidas por la agencia para cada viaje.

10.7. Resumen

En este capítulo se ha estudiado un caso de aplicación que se refiere a una compañía de autobuses. Al final, es apropiado hacer algunas consideraciones. Primero, se decidió describir los requerimientos *tal como son*, sin introducir cambios en los mismos. Así, los requerimientos (particularmente las estructuras de los procesos) reflejan claramente el funcionamiento de la organización antes de la introducción de un sistema de información automatizado. El nuevo sistema consistirá en una sola base de datos integrada, en lugar de almacenes de datos individuales, parcialmente automatizados. Esto sin duda introducirá modificaciones en la organización del trabajo. Por ejemplo, gracias al nuevo sistema será posible aceptar reservas durante las dos últimas horas antes de la salida, y uno de los contratiempos del sistema actual no existirá más.

Las modificaciones mayores pueden implicar la reorganización de los procesos, por ejemplo, eliminar la necesidad del proceso de GESTION_DE_HORARIO, ya que otros procesos pueden cambiar directamente la información de HORARIO en el computador. Como se observó al principio, los problemas de organización se consideran fuera del ámbito de este libro.

Otra consideración atañe a las **fronteras del sistema de información**. Si se consideran cuidadosamente los requerimientos, se descubrirá que se han omitido algunos elementos, como la compra, venta o reparación de los autobuses. Esto sucede porque durante el proceso de modelado del sistema de información, sus fronteras se hacen más claras. Por consiguiente, se decidió no introducir en ninguno de los esquemas, ni en el D ni en el F, las descripciones de aquellas porciones de los requerimientos que, en un punto dado del diseño, fueron consideradas fuera de esta frontera.

Al comparar el esquema D y el esquema F al final del proceso, está más que claro que el primero es un documento mucho más maduro, pues da la estructura exacta del esquema conceptual de datos; el segundo identifica los procesos

del sistema de información, pero no es del todo específico en sus procedimientos. Recuérdese que es deseable no llevar el análisis funcional más allá de un punto en que se torne procedural. De hecho, la distancia entre la identificación del proceso y la especificación de procedimiento es bastante amplia; tal distancia se reduce parcialmente al proveer esquemas de navegación para cada operación.

Ejercicio

- 10.1. Use las técnicas presentadas en este estudio de caso para diseñar el sistema de información de la biblioteca del departamento de informática que se describe. Haga suposiciones adicionales cuando se necesite y expóngalas explícitamente. Producza al final esquemas D y F, con la lista de las principales operaciones y sus esquemas de navegación.

En la biblioteca de un departamento de informática, los investigadores y los estudiantes pueden comprar libros. Los investigadores deben indicar la subvención que usan para comprar el libro; cada estudiante tiene un presupuesto limitado, que es fijado cada año por el decano de la universidad.

Algunos libros se escogen entre aquellos que envían periódicamente las compañías editoras para revisión; cada libro se solicita llenando una solicitud impresa. Cuando un libro para revisión es escogido, se conserva en la biblioteca y se emite una orden de compra; si los libros para revisión no son escogidos, se devuelven después de un corto lapso. Cuando un libro ordenado no es para revisión, se verifica la corrección de la solicitud contra el directorio de los libros recientemente publicados, y entonces se emite una orden. Cuando el bibliotecario recibe la factura (que viene con el libro en el segundo caso), su contenido se compara con la orden, y la factura se paga si no hay errores. En este punto, el bibliotecario pide al solicitante 10 palabras clave y la clasificación de acuerdo con las categorías ACM.

Junto con los libros, también se guardan en la biblioteca publicaciones periódicas. Todos los años, cada profesor del departamento puede expresar diez preferencias entre las revistas que se publican. Se escogen las veinte publicaciones con más alta puntuación.

A los estudiantes y profesores del departamento se les puede prestar libros. Los profesores también pueden pedir prestadas publicaciones periódicas. Los estudiantes no pueden tener más de cinco libros al mismo tiempo y no pueden tener un libro dado durante más de un mes. Cuando un estudiante conserva un libro más de un mes, se le envía una carta. Si esto ocurre dos o más veces en el mismo año, sólo se le prestarán más libros si el estudiante encuentra profesor que actúe como aval. Los profesores del departamento pueden conservar un número cualquiera de li-

bros y revistas, pero si quieren quedarse con un libro por más de un mes, tienen que decir dónde lo tienen y permitir al bibliotecario recuperarlo en cualquier momento.

Si un estudiante o un profesor pregunta por un libro que está prestado, se informa al solicitante la fecha en que el libro debe ser devuelto por quien lo ha tomado prestado. La solicitud es archivada con propósitos estadísticos: cuando un libro no está disponible para un gran número de solicitudes, se compra automáticamente un nuevo ejemplar.

Algunas veces, los libros desaparecen de la biblioteca, lo que se detecta periódicamente cuando se hace un inventario de los libros físicamente presentes en la biblioteca y se combina con la lista de los libros prestados, registrada en el directorio principal. Cada seis meses se publica una lista de los libros perdidos. Si no son devueltos en un mes, automáticamente se compran de nuevo.

Cuando un libro tiene 10 años, el director de la biblioteca decide si guardar o no el libro en el almacén de la biblioteca, en lugar de dejarlo en los estantes. Las publicaciones periódicas que no se han usado o prestado en un año no se compran el año siguiente.

Tercera parte

Diseño lógico y herramientas de diseño

El objetivo del diseño lógico es convertir el esquema conceptual de datos en un esquema lógico ajustado al sistema de gestión de base de datos específico que se tenga disponible. Mientras que el objetivo fundamental del diseño conceptual es lograr la compleción y expresividad del esquema, el diseño lógico pretende obtener una representación que use los recursos para la estructuración de datos y modelado de restricciones disponibles en el modelo lógico de la manera más eficiente posible.

En la tercera parte se considera el proceso de transformación donde el modelo conceptual es el modelo ER en su forma mejorada, como se ha visto hasta ahora, y el modelo objetivo es un modelo lógico comúnmente usado en los DBMS (*Data Base Management Systems*, sistemas de gestión de base de datos) de hoy en día. En la actualidad hay muchos partidarios y muy buena comprensión de los rasgos básicos de los sistemas relacionales, de redes y jerárquicos, y se han implantado miles de bases de datos que utilizan estos tres modelos. Por consiguiente, se considerarán los tres modelos de datos correspondientes como objetivos del diseño lógico. La fase del diseño físico sigue al diseño lógico y está estrechamente relacionada con las estructuras físicas de datos y de acceso disponibles en un DBMS en particular.

El capítulo 11 introduce un enfoque global del diseño lógico, distinguiendo dos fases: independiente del modelo y dependiente del modelo. El capítulo 11 se dedica al diseño lógico independiente del modelo, mientras que los próximos tres capítulos se dedican al diseño lógico dependiente del modelo para los tres modelos antes mencionados. El diseño lógico independiente del modelo está regulado por una estimación de la carga de la aplicación, por lo que se

desarrolla primero una metodología para captar esta información de carga. Sobre la base de los accesos anticipados a los datos, se analiza cómo simplificar el esquema conceptual eliminando jerarquías de generalización, dividiendo entidades e interrelaciones, o uniéndolas. También se analiza la selección de claves primarias.

Los capítulos 12, 13 y 14 se dedican al diseño lógico de bases de datos relacionales, de redes y jerárquicas, respectivamente. En cada capítulo se revisan, primero, el modelo de datos con sus rasgos de modelado, gestión de restricciones y enfoque de tratamiento de datos. Luego se analiza el diseño lógico en términos de transformación del esquema ER en el esquema correspondiente de ese modelo de datos. El diseño lógico comprende, además, las correspondencias de las especificaciones de alto nivel de las consultas y aplicaciones. Brevemente se analiza la correspondencia de esquemas de navegación con el lenguaje de consulta apropiado o los DML de los modelos.

Una característica común de estos tres modelos lógicos es la falta de ciertos rasgos de abstracción que se usan en los modelos conceptuales, como la generalización de entidades, abstracción de interrelaciones y ciertas restricciones, como las de cardinalidad. En consecuencia, es muy difícil para los diseñadores de bases de datos y para quienes desarrollan nuevas aplicaciones entender los datos ya existentes en las bases de datos implantadas que han evolucionado durante cierto tiempo. Por tanto, es necesaria una metodología de retroingeniería que produzca un esquema conceptual abstracto a partir de la base de datos ya implantada. Se muestra una metodología, paso a paso, para traducir esquemas lógicos existentes de los tres modelos al modelo ER.

Esta parte es una preparación para el capítulo 15, que estudia las herramientas de diseño de base de datos para el diseño lógico y conceptual.

Diseño lógico de alto nivel usando el modelo de entidades-interrelaciones

El objetivo del diseño lógico es convertir el esquema conceptual de datos en un esquema lógico ajustado específicamente al sistema de gestión de base de datos que se tenga a disposición. Mientras que el objetivo fundamental del diseño conceptual es la compleción y expresividad del esquema, el objetivo del diseño lógico es obtener una representación que use de la manera más eficiente posible los recursos para estructurar datos y modelar restricciones disponibles en el modelo lógico.

Se considerará el proceso de conversión para el cual el modelo conceptual es el modelo ER en su forma mejorada, tal como hasta ahora se ha visto, y el modelo objetivo será uno de los modelos lógicos comúnmente usados en los actuales sistemas de gestión de bases de datos comerciales, que incluyen los siguientes: relacional, de redes y jerárquico. El modelo orientado a objetos también se está popularizando; sin embargo, no existe un modelo estándar orientado a objetos. Por ello, en el momento de escribirse este libro consideramos prematuro abordar la transformación al modelo orientado a objetos de manera definitiva. Dentro de pocos años los modelos orientados a objetos tendrán varias implantaciones estables y es probable que surja un consenso. En comparación, existe un entendimiento muy bueno de los rasgos básicos de los sistemas relationales, de redes y jerárquicos, y miles de bases de datos ya han sido puestas en práctica. Estos tres planteamientos de modelado de datos se repasarán en los capítulos 12, 13 y 14, respectivamente.

Los tres modelos tienen ciertas similitudes: por ejemplo, se puede ver el modelo jerárquico como un caso especial del modelo de redes, y los sistemas de archivos convencionales y el modelo relacional tienen una estructura común plana de archivos y relaciones. La característica común de todos estos modelos es la falta de ciertos rasgos de abstracción que se usan en los modelos conceptuales, como generalización, abstracción de interrelaciones, y ciertas restricciones, como la de cardinalidad. En consecuencia, un aspecto importante del pro-

ceso de diseño lógico se refiere a la conversión de esos mecanismos de representación de alto nivel en términos de las estructuras de bajo nivel disponibles en el modelo lógico.

Este capítulo introduce un planteamiento global del diseño lógico que distingue dos fases: independiente del modelo y dependiente del modelo. El presente capítulo se dedica al diseño lógico independiente del modelo; los próximos tres capítulos se refieren a los diseños lógicos dependientes del modelo para los tres modelos arriba mencionados. Los objetivos generales de este capítulo son: 1) desarrollar una metodología para estimar las cargas de aplicación en términos de los volúmenes de datos y de los requerimientos de proceso, 2) simplificar el esquema conceptual ER con base en la información de carga de la aplicación, y 3) hacer recomendaciones para las decisiones de diseño lógico independiente del modelo.

Los puntos segundo y tercero dependen en forma crucial de la disponibilidad de información cuantitativa sobre volúmenes de datos y requerimientos de proceso.

En este capítulo se intenta ilustrar el proceso, sin pretender presentar un análisis cuantitativo exhaustivo. El apartado 11.1 describe más detalladamente el proceso global del diseño lógico. El apartado 11.2 muestra cómo se puede modelar la carga de la base de datos, tanto para datos como para operaciones. En el apartado 11.3 se analizan las decisiones de diseño que se refieren a los datos derivados, dando criterios generales sobre qué conservar en el esquema. El apartado 11.4 trata las jerarquías de generalización y subconjuntos, mostrando cómo pueden estar representados en términos de entidades e interrelaciones para así simplificar las correspondencias subsecuentes. Los apartados 11.5 y 11.6 describen dos tipos de transformaciones que se pueden aplicar a las entidades e interrelaciones para obtener un esquema eficiente: primero, dividiendo las entidades (e interrelaciones) en grupos y luego fusionando grupos de entidades e interrelaciones en una sola entidad. El apartado 11.7 explica cómo se puede seleccionar la clave primaria entre los identificadores de una entidad. Finalmente, en el apartado 11.8 se continúa con el caso de estudio del capítulo anterior; se hacen ciertas suposiciones sobre los volúmenes de datos y la naturaleza esperada del procesamiento, y se aplica la metodología general descrita en este capítulo.

11.1. Un enfoque global del diseño lógico

La figura 11.1 muestra el proceso global del diseño lógico y sus entradas y salidas. En lugar de tratar el diseño lógico como una sola actividad, se ha preferido utilizar un planteamiento modular, en el cual se distinguen dos fases: diseño lógico de alto nivel y diseño lógico dependiente del modelo. El primero es una etapa independiente del modelo, común a todos los modelos de DBMS, mientras que la etapa dependiente del modelo realiza una correspondencia específica.

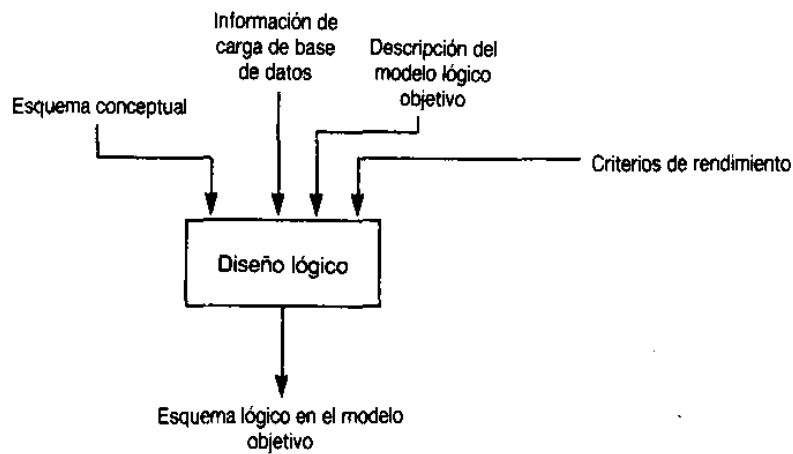


Figura 11.1. Entradas y salidas en el proceso de diseño lógico.

El diseño lógico dependiente del modelo lleva el esquema modificado derivado de la primera fase a un modelo de datos objetivo específico.

Obsérvese cómo las diferentes entradas agrupadas en la figura 11.1 se han separado en la figura 11.2. La información acerca de la carga de base de datos guía la primera fase, en la cual el esquema conceptual se modifica para tener

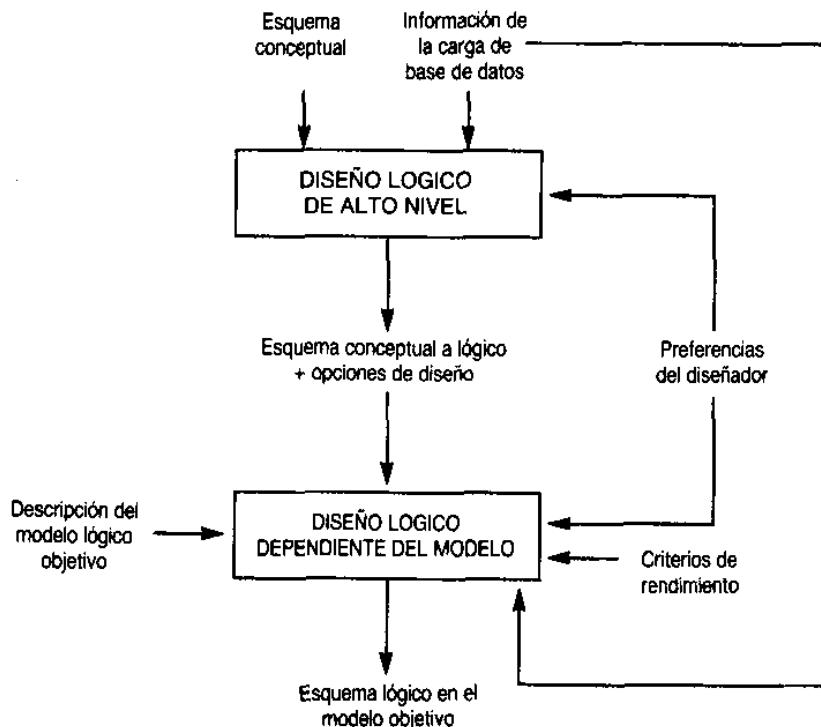


Figura 11.2. Un planteamiento de dos fases del diseño lógico.

una versión más simplificada. En caso de haber diferentes opciones, se le puede pedir al diseñador que use su propia preferencia. La etapa dependiente del modelo la regulan los rasgos disponibles en el modelo de datos objetivo, y de nuevo deja margen para las elecciones del diseñador. Se han mostrado las preferencias del diseñador como una entrada de las dos fases del diseño lógico. Sin embargo, el impacto de las decisiones de diseño tomadas en estas dos fases sobre el rendimiento es muy difícil de cuantificar, por lo que sólo sirven como pautas.

Este capítulo trata el diseño lógico independiente del modelo, cuyo objetivo es ejecutar actividades de transformación y optimización del esquema conceptual. El resultado de esta etapa es un esquema intermedio «conceptual a lógico», que se puede ver como una versión simplificada y parcialmente optimizada del esquema de entrada. Se llama al resultado esquema *conceptual a lógico* porque representa detalles adicionales, como la selección de claves, la división de entidades, etc., que son lógicas más que conceptuales. Las entradas al diseño lógico (Figs. 11.1 y 11.2) son:

1. El esquema conceptual.
2. Una descripción del modelo lógico objetivo y las restricciones.
3. Datos de carga, es decir, la población de la base de datos y el conocimiento de consultas y transacciones que se realizan en la base de datos, junto con su frecuencia. Este conocimiento permite determinar la representación óptima del esquema en términos del modelo lógico.
4. Criterios de rendimiento provistos como requerimiento para la eficiencia de la implantación. Las medidas y restricciones típicas de rendimiento son las siguientes:
 - a) Tiempo de respuesta (máximo o promedio).
 - b) Almacenamiento ocupado por la base de datos.
 - c) Utilización de CPU o tiempo de E/S.
5. Preferencias del diseñador. Estas se recogen durante el proceso interactivo del diseño lógico.

Obsérvese que en la figura 11.2 las entradas 2 y 4 se alimentan a la segunda fase del diseño lógico. La entrada de preferencia del diseñador va en las dos fases del diseño lógico. La salida de la segunda etapa es el esquema lógico en el sistema objetivo, y se somete luego al proceso de diseño físico. El **diseño físico** se ocupa del diseño del esquema en un DBMS donde se han tomado en cuenta las selecciones del diseño en relación con índices, opciones de punteros, agrupamiento de registros, partición del almacenamiento de la base de datos, enlace de registros y uso de buffers. El diseño físico tiende a estar estrechamente unido a DBMS específicos y es guiado por criterios de rendimiento. En este libro no nos ocuparemos de él.

Dado que los modelos conceptuales se usan como herramienta de diseño independiente de los DBMS y tienden a estar alejados de los modelos lógicos, la

pregunta obvia es ¿cómo se puede tratar las consideraciones de rendimiento en la etapa del diseño independiente del modelo? *No se puede* decir que el mejoramiento del rendimiento se pueda alcanzar de manera definitiva por la utilización del esquema conceptual, porque ese esquema experimenta muchos niveles de decisiones de diseño adicionales antes de que el diseño físico esté completo. Aun con un diseño físico determinado, hay algunas variables que afectan el rendimiento definitivo: 1) la mezcla de aplicaciones (si las transacciones ocurren con cierta/amplia variabilidad); 2) la naturaleza de la optimización de consultas, del control de concurrencia, y de los algoritmos de recuperación en el DBMS; 3) otras aplicaciones ejecutadas dentro del mismo sistema, y así sucesivamente.

Con estos antecedentes, nuestros objetivos permanecen como se manifestaron al principio. Se desea dar cuenta de la carga principal de la base de datos seguir la cual el sistema debe desempeñarse bien. Para lograr eso, se mide el efecto de las transacciones importantes considerándolas en términos de los esquemas de navegación. En concreto, se considera la estructura de las operaciones, su frecuencia y el número de casos de entidades e interrelaciones visitadas por las operaciones. El objetivo es realizar algunas transformaciones obvias que ayudarán al subsecuente diseño lógico dependiente del modelo. Adicionalmente, se desea transformar, durante esta fase, las construcciones más ricas, como la jerarquía de generalización, en entidades e interrelaciones, de manera que las correspondencias posteriores sean claras.

11.2. Modelado de la carga de bases de datos

Por *carga* de la base de datos se entiende las actividades o aplicaciones que se pide ejecutar a la base de datos. Se usa el término *carga* de una manera similar a como se usa en los sistemas de ingeniería, donde se describe la carga de una viga (ingeniería civil), la carga de un circuito (ingeniería eléctrica) o la carga en un transportador (ingeniería mecánica). Para caracterizar la carga apropiadamente, se usan dos tipos de información: el volumen de datos y la descripción de las aplicaciones.

El volumen de datos se mide en el modelo ER determinando los siguientes parámetros: 1) el número promedio $N(E)$ o $N(R)$ de casos de cada entidad E o interrelación (R), y 2) la cardinalidad promedio, card-prom (E, R) de cada entidad E en cada interrelación R. La población promedio (valores distintos) de un atributo puede ser diferente de la población de la entidad o interrelación correspondiente por dos razones. Primera, el atributo puede tener cardinalidades diferentes de (1,1). En el ejemplo de la figura 11.3 todos los atributos del esquema tienen el mismo número de ocurrencias que la entidad o interrelación correspondiente, excepto NUMERO_TELEFONO, para el cual se supone una proporción de 1,5 teléfonos por empleado. Por ende, su cardinalidad es 1,5 veces la cardinalidad de la entidad CLIENTE. Segunda, puede haber muchos valores duplica-

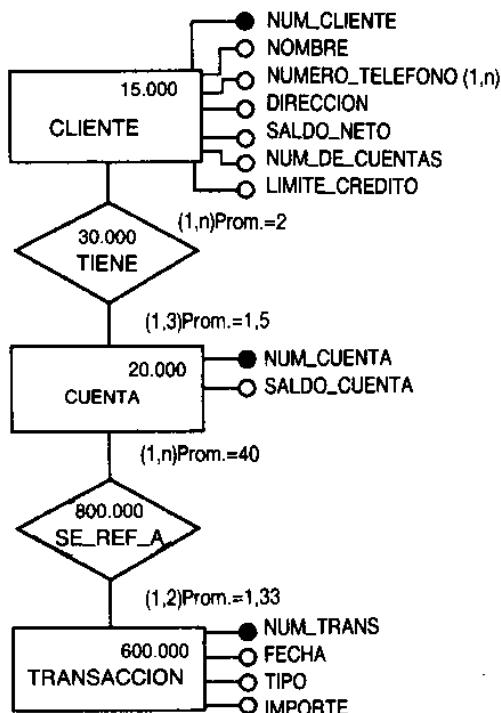


Figura 11.3. Un ejemplo de esquema con información de volumen de datos.

dos para un atributo. Por ejemplo, el atributo TIPO de TRANSACCION tiene una cardinalidad de 10, suponiendo que hay solamente 10 tipos distintos de transacciones. El atributo SALDO CUENTA tendrá una cardinalidad más baja que la de la entidad CUENTA, debido a valores idénticos de los saldos. Aun así, se puede suponer que el promedio estimado sea el mismo que para la entidad CUENTA. Idealmente, el promedio estimado del número de ocurrencias de un atributo indica el número de valores distintos de esa estimación.

Dado un esquema (Fig. 11.3), la información de volumen de datos puede estar representada en el esquema mismo con cardinalidades inmersas dentro de las entidades e interrelaciones, y cardinalidades promedio de las interrelaciones mencionadas junto con las cardinalidades mínimas y máximas. El esquema representa una base de datos con un promedio de 15.000 clientes, 20.000 cuentas y 600.000 transacciones. Para cada cliente hay, en promedio, 2 cuentas; para cada cuenta, un promedio de 1,5 clientes y 40 transacciones; cada transacción está relacionada con un promedio de 1,33 cuentas. Esto es porque algunas transacciones implican múltiples cuentas (por ejemplo, transferencias de la cuenta de cheques a la de ahorro, de la de ahorro a la de préstamos, etc.).

Obsérvese que para cualesquiera dos entidades E_1 y E_2 y para cualquier interrelación binaria R entre ellas,

$$N(E_1) \times \text{card-prom}(E_1, R) = N(E_2) \times \text{card-prom}(E_2, R) = \text{card-prom} \quad (1)$$

Tabla 11.1. Tabla de volumen de datos

Concepto	Tipo	Volumen
CLIENTE	E	15.000
CUENTA	E	20.000
TRANSACCION	E	600.000
TIENE	R	30.000
SE_REFIERE_A	R	800.000
NUM_CLIENTE	A	15.000
NOMBRE	A	15.000 ^a
NUMERO_TELEFONO	A	22.500 ^b
DIRECCION	A	15.000
SALDO_NETO	A	15.000 ^a
NUM_DE CUENTAS	A	10
LIMITE_CREDITO	A	50
NUM CUENTA	A	20.000
SALDO CUENTA	A	20.000 ^a
NUM_TRANS	A	600.000
FECHA	A	730 ^c
TIPO	A	10
IMPORTE	A	600.000 ^a

^a Estimación en el peor de los casos^b Indica atributo polivalente^c Supone datos de transacciones de dos años solamente

Como alternativa, esta información puede colocarse en una tabla de volumen de datos. La tabla 11.1 es una relación de este tipo que se refiere al esquema de la figura 11.3.

La carga de aplicación se estima en términos de las operaciones importantes (transacciones por lotes y en línea, así como consultas *ad hoc*). Para la mayoría de las situaciones prácticas, es imposible tener una idea precisa de la carga futura de una base de datos. En consecuencia, se debe tratar de estimar la carga en términos de las operaciones usadas más comúnmente. En el capítulo 9 se modelaron las operaciones en términos de esquemas de navegación. Cada operación es descrita por lo siguiente:

1. El esquema de navegación para la operación, tal como se describe en el apartado 9.4. El esquema de navegación incluye los elementos (entidades, atributos, interrelaciones) del esquema conceptual que son usados por la operación, e indica la secuencia (o la dirección) en la que se obtiene acceso a cada elemento, denominada *navegación* en la base de datos.
2. Para cada entidad o interrelación en el esquema de navegación:
 - a) Una indicación del tipo de acceso realizado a la entidad o interrelación, distinguiendo los accesos de lectura (recuperar) y escritura (insertar, actualizar, borrar).

- b) El número promedio de casos de la entidad o interrelación que son usados por la operación.
3. La frecuencia de activación promedio de una operación, medida en unidades apropiadas (por ejemplo, 10 veces al día, 5 veces al mes).
 4. El tipo de la operación, esto es, si es realizada por lotes o en línea.

Se resume la información anterior en dos cuadros denominados tabla de frecuencia de operaciones y tabla de volumen de acceso de operaciones. La tabla de frecuencia de operaciones tiene columnas para: 1) el nombre de la operación (un nombre abreviado para la transacción), 2) la descripción (un comentario breve sobre la operación), 3) la frecuencia (indica el número de veces que se ejecuta esta operación, en promedio), y 4) el tipo (indica si la operación ocurre por lotes [PL] o en línea [EL]). La tabla de volumen de acceso de operaciones se construye para cada operación de base de datos (transacción o consulta) e incluye las siguientes columnas: 1) nombre de la operación; 2) nombre del concepto (se refiere a los conceptos a los que se tiene acceso en el esquema de navegación); 3) tipo de concepto (E/R: E para entidad, R para interrelación); 4) lectura/escritura (se refiere a si es un acceso de lectura o de escritura [insertar/borrar/actualizar]); y 5) promedio de ocurrencias a las que se tuvo acceso.

Para el ejemplo del banco que se expuso anteriormente, se puede considerar las siguientes operaciones:

- O1: ABRIR UNA CUENTA PARA UN CLIENTE NUEVO.
- O2: LEER EL SALDO NETO DE UN CLIENTE.
- O3: MOSTRAR LAS ULTIMAS 10 TRANSACCIONES DE UNA CUENTA.
- O4: RETIRAR DINERO DE UNA CUENTA.
- O5: DEPOSITAR DINERO EN UNA CUENTA.
- O6: PREPARAR UN ESTADO MENSUAL DE LAS CUENTAS.
- O7: PARA UN CLIENTE DADO, INFORMAR EL NUMERO DE CUENTAS QUE TIENE.
- O8: PARA UN CLIENTE DADO, MOSTRAR LA FECHA, Y EL IMPORTE DE LA ULTIMA TRANSACCION RELACIONADA CON CUENTAS CUYO SALDO ES NEGATIVO.

La tabla 11.2 es la relación de frecuencia de operaciones para las operaciones arriba citadas. La tabla 11.3 es la relación de volumen de acceso de operaciones para las primeras cinco operaciones. Obsérvese que el análisis estará basado en *un día* como unidad de tiempo para procesamiento. O6 es una operación mensual por lotes y no entrará en el análisis. Los esquemas de navegación se dejan como ejercicio para el lector, el cual puede también practicar construyendo tablas hipotéticas de volumen de acceso de operaciones para las operaciones restantes.

Tabla 11.2. Tabla de frecuencia de operaciones

Nombre/descripción de la operación	Frecuencia	Tipo (en línea/ por lotes)
O1 ABRIR UNA CUENTA	100 veces al día	EL
O2 LEER EL SALDO	3.000 veces al día	EL
O3 MOSTRAR ULTIMAS 10 TRANSACCIONES	200 veces al día	EL
O4 RETIRAR DINERO	2.000 veces al día	EL
O5 DEPOSITAR DINERO	1.000 veces al día	EL
O6 PREPARAR UN ESTADO MENSUAL	1 vez al mes	PL
O7 INFORMAR NUM. DE CUENTAS QUE TIENE UN CLIENTE	75 veces al día	EL
O8 MOSTRAR TRANSACCIONES DE CUENTAS CON SALDO NEGATIVO	20 veces al día	EL

Antes de finalizar este apartado, queremos hacer varias observaciones. Primero, el esquema de navegación *no representa* una especificación completa de la información necesaria para tomar todas las decisiones del diseño lógico independiente del modelo. Los predicados de selección exactos, las listas de atributos que se obtendrán de las entidades e interrelaciones, etc., no están completamente especificadas.

Segundo, se indica si una operación se realiza en línea o por lotes en la tabla de frecuencia de operaciones. Las operaciones en línea deben ser consideradas más cruciales así como más costosas que las operaciones por lote. Para evitar detalles excesivos, no se muestra aquí cómo se debe dar cuenta de esta diferencia cuantitativamente más que cualitativamente.

Tercero, recoger tanta información sobre la carga de aplicación es difícil, especialmente para aplicaciones de base de datos amplias. Por otro lado, es necesario conocer la carga de aplicación para tomar decisiones durante los diseños lógico y físico que influyen en el rendimiento final del sistema. En la práctica, las operaciones siguen la llamada regla de 20-80, establecida como observación empírica en la mayoría de las situaciones de procesamiento de volúmenes grandes: el 20% de las operaciones produce el 80% de la carga. En consecuencia, nos debemos concentrar por lo menos en el 20% más importante de las operaciones. Ahora se procederá con las decisiones del diseño lógico independientes del modelo.

11.3. Decisiones sobre datos derivados

Las entidades o interrelaciones de un esquema algunas veces contienen atributos derivados, es decir, atributos que se pueden calcular realizando procedi-

Tabla 11.3. Tabla de volumen de acceso de operaciones

Nombre/descripción de la operación	Concepto	Tipo concreto	Lectura/escritura	Promedio ocurrencias a las que se tuvo acceso
O1 ABRIR UNA CUENTA	CUENTA	E	E	100
	CLIENTE	E	E	$100 \times 1,5 = 150$
	TIENE	R	E	$100 \times 1,5 = 150$
O2 LEER EL SALDO	CUENTA	E	L	3.000
O3 MOSTRAR LAS 10 ULTIMAS TRANSACCIONES	CUENTA SE_REFIERE_A TRANSACCIONES	E R E	L L L	200 $200 \times 40 = 8.000$ seleccionar 2 000 de 8.000
O4 RETIRAR DINERO	CUENTA	E	L	2.000
	CLIENTE	E	E	2.000
O5 DEPOSITAR DINERO*	CUENTA CLIENTE	E E	L E	1.000 1.000 $1.000 \times 1,5 = 1500$
O6 PREPARAR ESTADO MENSUAL				
O7 INFORMAR NUM. DE CUENTAS QUE TIENE UN CLIENTE				
O8 MOSTRAR TRANSACCIONES DE CUENTAS CON SALDO NEGATIVO				

Nota: Véase el ejercicio 11.2 para completar las últimas tres filas.

* Suponga que también se actualiza SALDO_NETO en CLIENTE.

mientos matemáticos con valores de otros atributos, contando los casos de alguna entidad relacionada, etcétera. Mantener datos derivados en el esquema lógico tiene una ventaja y varias desventajas. La ventaja es que no se necesita calcular el valor de ese dato durante la ejecución; por consiguiente, se reduce el número de accesos a la base de datos. Las desventajas son dos: en primer lugar, se requiere procesamiento adicional para mantener los datos derivados congruentes con los otros datos relacionados en los que se basa. Cada vez que una actualización afecte un valor derivado, éste deberá ser calculado de nuevo. En segundo lugar, se requiere una mayor cantidad de almacenamiento.

La decisión sobre mantener los datos derivados o retirarlos del esquema y calcularlos según se requiera debe basarse en un balance entre los pros y contras antes mencionados. Todas las operaciones de recuperación que necesitan los datos derivados se benefician si éstos se conservan; todas las operaciones de escritura que actualizan cualesquiera valores de datos a partir de los cuales se calculan los datos derivados representan un tiempo de procesamiento adicional.

El almacenamiento que se requiere para mantener los datos derivados también representa una carga extra. Así, la decisión respecto a los datos derivados debe estar basada en una evaluación de estos dos grupos de consideraciones opuestas. Este proceso se ilustra en seguida con el ejemplo del banco.

Por ejemplo, considérese el atributo SALDO_NETO de la entidad CLIENTE de la figura 11.3, que se puede calcular sumando el atributo SALDO CUENTA de todas las cuentas (ocurrencias de la entidad CUENTA) que tiene un cliente. Para decidir si mantener o no este atributo en el esquema, necesitamos saber con qué frecuencia se tiene acceso a él y cuán a menudo se vuelve a calcular debido a cambios en el atributo SALDO CUENTA de cada cuenta. Considérense los puntos que siguen:

1. Las operaciones O1 (ABRIR UNA CUENTA), O3 (MOSTRAR LAS ULTIMAS 10 TRANSACCIONES) y O6 (PREPARAR UN ESTADO MENSUAL) no usan el atributo SALDO_NETO.
2. Beneficio: La operación O2 (LEER SALDO) se beneficia al tener el atributo SALDO_NETO; si se retira del esquema, O2 tendrá que visitar la interrelación TIENE y la entidad CUENTA un promedio de 6.000 (3.000×2) veces¹. El beneficio se obtiene solamente si el atributo se mantiene actualizado todo el tiempo.
3. Costo. Las operaciones O4 (RETIRAR DINERO) y O5 (DEPOSITAR DINERO) requieren un tiempo de procesamiento adicional; para mantener SALDO_NETO, realizan respectivamente $3.000 = (2.000 \times 1,5)$ y $1.500 = (1.000 \times 1,5)$ accesos a la interrelación TIENE y a la entidad CLIENTE². Esto representa 4.500 accesos extra, así como 4.500 operaciones de escritura.
4. El requerimiento adicional de almacenamiento es de alrededor de 90 kilobytes = $15.000 \text{ clientes} \times 6 \text{ bytes por cliente}$.

Por consiguiente, hay que escoger entre el beneficio de los accesos ahorrados y el costo de procesamiento y almacenamiento adicional. Según los números anteriores, se debe retirar del esquema el atributo derivado SALDO_NETO; aunque la memoria para los datos derivados por cliente es insignificante, el costo de procesamiento es de 4.500 operaciones de escritura para ahorrar un promedio de 1.500 ($=6.000 - 4.500$) accesos adicionales. Esto favorece la eliminación; en general, una escritura es mucho más costosa que una recuperación. Obsérvese que la decisión anterior puede cambiar si otras transacciones importantes entran en consideración e inclinan la balanza en favor del almacenamiento, o si algunas frecuencias cambian; por ejemplo, si O2 se realiza 6.000 veces al día.

Se considerará ahora otro dato derivado, NUM_DE CUENTAS de la entidad CLIENTE, que calcula el número total de cuentas (un entero de 2 bytes) que tiene un cliente.

1. Las operaciones O2 a O6 y O8 no usan el atributo NUM_DE CUENTAS.

¹ Obsérvese que estamos suponiendo que el acceso a CUENTA a través de TIENE es un solo acceso.

² Obsérvese que estamos suponiendo que el acceso a CLIENTE a través de TIENE es un solo acceso.

2. Beneficio: La operación O7 (INFORMAR EL NUMERO DE CUENTAS QUE TIENE UN CLIENTE) se beneficia de tener el atributo NUM_DE_CUENTAS; si se retira del esquema, O7 tendrá que visitar la interrelación TIENE y la entidad CUENTA un promedio de 150 (75×2 cuentas por cliente) veces.
3. Costo. La operación O1 (ABRIR UNA CUENTA) requiere un tiempo de procesamiento adicional; para mantener NUM_DE_CUENTAS, necesita realizar 150 ($=100$ cuentas $\times 1,5$ clientes por cuenta) accesos a la entidad CLIENTE. Para estos clientes, se requieren 300 ($=150$ clientes $\times 2$ cuentas por cliente) accesos a la interrelación TIENE y a la entidad CUENTA para poder calcular el atributo NUM_DE_CUENTAS de CLIENTE. Así pues, hay 150 operaciones de escritura, una por cliente.
4. El requerimiento adicional de almacenamiento es de alrededor de 30 kilobytes = 15.000 clientes $\times 2$ bytes por cliente.

Aquí el costo de mantener el atributo derivado NUM_DE_CUENTAS ciertamente sobrepasa el beneficio, pues se tiene un ahorro de 150 accesos contra el costo de 450 accesos y 150 operaciones de escritura. Además, el gasto adicional de los 30 kilobytes de almacenamiento también se evita al retirar el atributo NUM_DE_CUENTAS del esquema.

11.4. Eliminación de jerarquías de generalización

Los modelos lógicos, incluidos los modelos relacionales, jerárquicos y de redes, no permiten representar las jerarquías de generalización ni los subconjuntos. En consecuencia, se deben modelar las jerarquías de generalización y los subconjuntos usando sólo entidades e interrelaciones. Al hacer esto debemos cuidar dos cosas: 1) la herencia de atributos de la entidad principal a la entidad subconjunto se debe indicar explícitamente³, y 2) la interrelación implícita ES_UN, que modela el hecho de que la entidad subconjunto es un caso especial de la entidad principal, se debe también captar. Se puede representar mediante una interrelación o usando un atributo para designar la subentidad apropiada en cuestión.

Para facilitar el análisis, llamaremos a las entidades implicadas *superentidad* y *subentidad*, con sus significados obvios. El planteamiento consiste en escoger entre las tres alternativas que mencionaremos en seguida. Lo ideal es que la selección la determine un análisis de la carga del esquema; nos abstendremos de hacer un análisis generalizado de esta selección, pero se señalarán algunas reglas generales con un simple ejemplo. Las tres alternativas son:

1. Integrar la jerarquía de generalización en una sola entidad uniendo los atributos de las subentidades y añadiendo estos atributos a los de la superentidad.

³ En el capítulo 2 se llamó a la entidad principal *entidad genérica*.

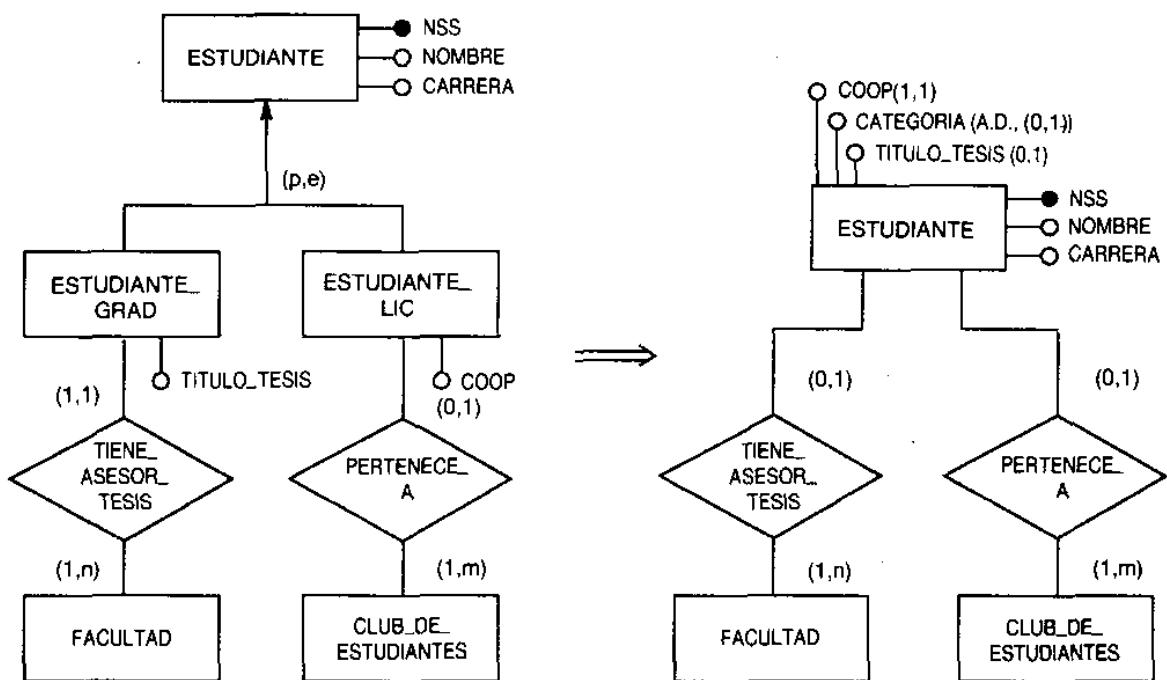


Figura 11.4. Modelado de una jerarquía de generalización mediante una superentidad.

dad. Esto se permite si la distinción entre las subentidades no es significativa desde el punto de vista de una aplicación⁴. Más aún, se añade un atributo discriminativo para indicar el caso al cual pertenece la entidad en consideración.

2. Eliminar la superentidad pero retener las subentidades. Aquí, los atributos heredados deben propagarse entre las subentidades.
3. Retener todas las entidades y establecer explícitamente las interrelaciones entre la superentidad y las subentidades.

Estas tres opciones se ilustrarán con algunos ejemplos y también se mostrarán los efectos de la transformación sobre las interrelaciones relacionadas.

11.4.1. Jerarquía de generalización modelada por la entidad superconjunto

En el esquema de jerarquía de generalización de la figura 11.4 la superentidad ESTUDIANTE tiene dos subentidades, ESTUDIANTE_GRAD y ESTUDIANTE_LIC, que designan los estudiantes graduados y de licenciatura, respectivamente. Suponiendo que esta distinción no sea pertinente para la aplicación más importante, se procederá a representar a todos los estudiantes con la única entidad ESTU-

⁴ Esto puede parecer contradictorio, porque si la distinción entre las subentidades no fuera significativa no debería estar modelada en el esquema conceptual en primer lugar. No obstante, se puede considerar razonable si el diseño lógico está regulado sólo por aplicaciones *importantes*, y esas aplicaciones no hacen la distinción.

DIANTE (denominada *entidad objetivo*). La unión de los atributos de las subentidades, a saber, COOP (estudio cooperativo) y TITULO_TESIS, se añade a la entidad objetivo. Los atributos pertenecientes a las subentidades se tratan como opcionales; en consecuencia, su cardinalidad es (0,1). Un atributo discriminativo (AD) CATEGORIA se añade a la entidad objetivo para distinguir los estudiantes graduados y los de licenciatura. La cardinalidad del AD es (1, 1) en el caso de las generalizaciones totales y exclusivas; la cardinalidad mínima del AD es 0 si la generalización es parcial, y su cardinalidad máxima es n si la generalización es superpuesta; aquí una entidad puede pertenecer a más de una subentidad⁵.

Las desventajas de la alternativa 1 son dos. Primero, puede generar un gran número de valores nulos para los atributos que se aplican sólo a las subentidades. Por ejemplo, en el esquema de la figura 11.4, se tienen valores nulos para los atributos no aplicables: COOP para los estudiantes graduados y TITULO_TESIS para los de licenciatura. En los ejemplos de la vida real puede haber decenas o centenas de estos atributos. Segundo, todas las operaciones que tenían acceso sólo a subentidades tienen ahora que buscar los casos correspondientes dentro del conjunto completo de casos de la superentidad.

Las ventajas de este planteamiento son también dos. Primero, ésta es la solución más simple desde el punto de vista del esquema resultante: no se necesitan interrelaciones. Segundo, teóricamente, esta alternativa es aplicable a todos los tipos de jerarquías de generalización: total o parcial; no superpuesta o superpuesta.

La figura 11.4 también muestra cómo gestionar las interrelaciones existentes de las subentidades para la entidad objetivo. Así, las interrelaciones TIENE_ASESOR_TESIS y PERTENECE_A se retienen, y ahora se aplican a la entidad objetivo ESTUDIANTE, dependiendo si está graduado o no. Esto se refleja en el cambio de 1 a 0 en la cardinalidad mínima de TIENE_ASESOR_TESIS.

11.4.2. Jerarquía de generalización modelada por las entidades subconjunto

En la jerarquía de generalización de la figura 11.5 se muestran los empleados divididos en secretarias, ingenieros y gerentes. En este caso, se les modela de manera independiente con tres entidades objetivo distintas. Los atributos comunes NSS y NOMBRE se propagan a cada una. Este identificador original NSS de EMPLEADO se convierte en identificador de cada entidad objetivo. La alternativa 2 tiene varias desventajas:

1. Esta alternativa no es práctica para generalizaciones superpuestas o parciales (como la que se muestra en la figura 11.6); sólo es práctica para jerarquías totales y exclusivas.
2. Se pierde el concepto de que las subentidades originales son subconjuntos

⁵ Es posible idear un esquema de codificación en el que un solo valor asignado al atributo discriminativo puede valer para una combinación de subentidades.

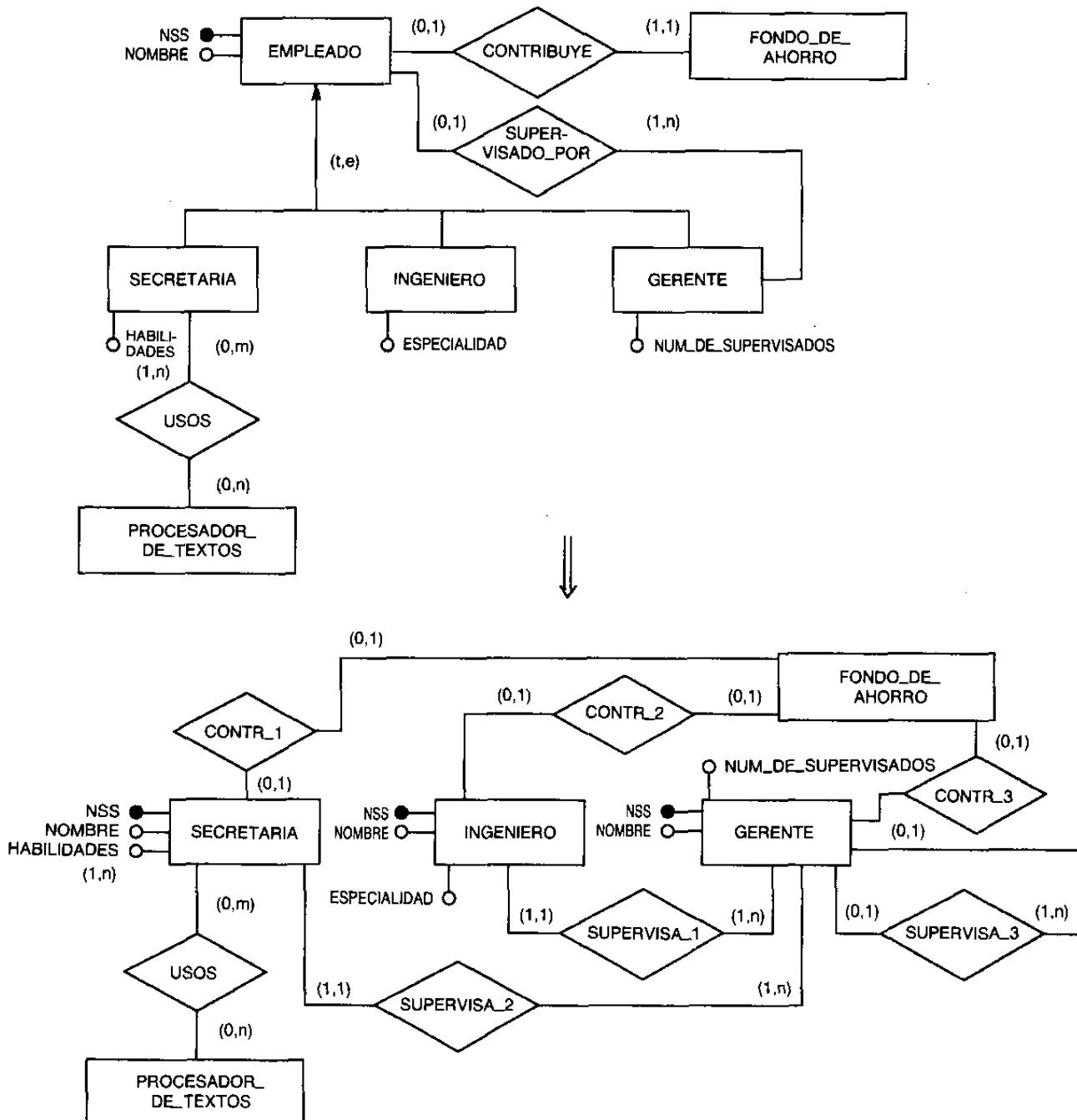


Figura 11.5. Modelado de una jerarquía de generalización mediante entidades subconjunto.

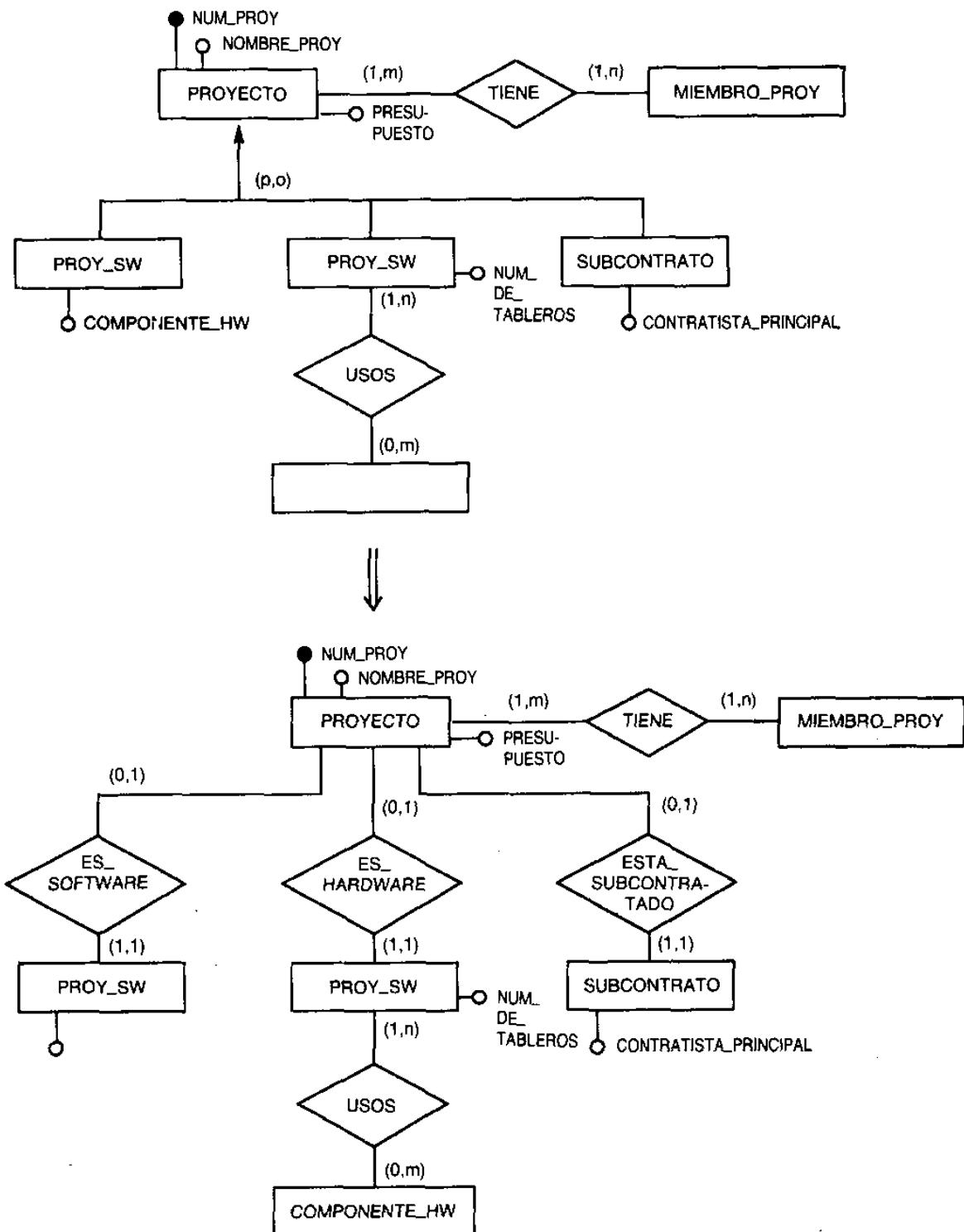


Figura 11.6. Modelado de una jerarquía de generalización mediante interrelaciones.

de la misma entidad. Así, en el ejemplo antes citado, se pierde la información importante de que las secretarias, ingenieros y gerentes son todos empleados. Esto puede ser vital para cierto procesamiento.

3. Si el número de atributos de la superentidad (comunes a todas las entidades) es excesivo, su duplicación en el esquema de cada subentidad no se justifica.
4. Cada operación que originalmente tenía acceso sólo a la superentidad queda en desventaja: debe obtener acceso ahora a todos los casos de todas las subentidades.

Otra desventaja aparece cuando se considera el modelado de las interrelaciones originales de la superentidad. Emplear una interrelación como usos de la figura 11.5, en la que únicamente participa una subentidad, no causa problema. En cambio, la interrelación SUPERVISADO_POR debe repetirse para cada subentidad; obsérvense las tres interrelaciones individuales que surgen. La interrelación CONTRIBUYE está también representada de tres maneras diferentes. Obsérvese que la cardinalidad mínima de FONDO_DE_RETIRO en cada una de las tres interrelaciones objetivo es 0 porque cada contribución está asociada sólo con un empleado. A nivel de esquemas, este uso repetitivo de la misma interrelación crea una redundancia no deseable.

Esta alternativa es ideal cuando el concepto común representado por la superentidad no se requiere en el diseño lógico y cuando los atributos comunes para las subentidades son muy pocos comparados con los atributos distintivos individuales. También resulta beneficiosa cuando las operaciones están restringidas a las subentidades.

11.4.3. Jerarquía de generalización modelada por interrelaciones

La alternativa 3 puede ser considerada como la más general de las tres: siempre es posible. Considérese el esquema de jerarquía de generalización de la figura 11.6, donde se muestra que los proyectos de una compañía de computadores se subdividen en proyectos de software y de hardware, así como en proyectos subcontratados. La jerarquía es parcial y superpuesta. Así, en los dos casos extremos, un proyecto puede tener información dispersa entre las tres subentidades o en ninguna de ellas. La primera alternativa citada puede usarse; la segunda no. En la tercera alternativa, cada eslabón conjunto-subconjunto está explícitamente representado por una interrelación que es obligatoria (card-mín = 1) para la subentidad y opcional (card-mín = 0) para la superentidad. Cada entidad mantiene sus propios atributos; la identificación de las subentidades puede ser provista externamente por la superentidad o explícitamente añadida a cada subentidad.

En el esquema de la figura 11.6 todas las entidades originales y sus atributos se preservan en el esquema objetivo. Se añaden diferentes interrelaciones ES_UN para cada subentidad. Obsérvese que las interrelaciones que unen PROYECTO a MIEMBRO_PROY y PROYE_HW a COMPONENTE_HW también se preservan sin cambio en el esquema objetivo.

Esta alternativa tiene dos desventajas. 1) El esquema resultante es bastante complejo; por ejemplo, para insertar un nuevo caso de subentidad se requiere insertar dos casos adicionales: uno para la superentidad y otro para la interrelación con la superentidad. 2) Hay una redundancia inherente (al menos en el nivel conceptual) al representar cada eslabón ES_UN en la jerarquía original a través de una interrelación explícita.

La principal ventaja de la alternativa 3 es que modela las cuatro combinaciones de jerarquías parcial/total y exclusiva/superpuesta. Por consiguiente, es la más flexible en términos de cambios en los requerimientos de aplicación. Esta solución es conveniente también si la mayoría de las operaciones son estrictamente locales respecto a la superentidad o a una de las subentidades, de manera que sólo pocas operaciones pueden necesitar usarlas juntas.

11.4.4. Una metodología general para el empleo de las jerarquías de generalización

Se puede aplicar el criterio antes expuesto a fin de llegar a la siguiente estrategia general de toma de decisiones para utilizar las jerarquías de generalización. La alternativa 2, que usa sólo las subentidades, se rechaza de inmediato si la generalización no es total y exclusiva. De otra manera se decide entre las alternativas dividiendo en dos conjuntos todas las operaciones que usen entidades dentro de la generalización. Llamaremos a las subentidades E_1 y E_2 y la superentidad E_{12} (por ejemplo, E_{12} es EMPLEADO, E_1 es SECRETARIA, E_2 es INGENIERO). Los dos conjuntos de operaciones son:

1. El conjunto de operaciones que usan atributos de E_{12} sin importar la subdivisión de los casos de E_{12} en casos de E_1 y E_2 . Estas operaciones se facilitan si se modela la jerarquía de generalización exclusivamente con la superentidad (alternativa 1) o con interrelaciones (alternativa 3).
2. El conjunto de operaciones que usan una combinación de algunos atributos de la superentidad y otros sólo de la subentidad de E_1 o sólo de E_2 (por ejemplo, [E_{12} y E_1] o bien [E_{12} y E_2]). Estas operaciones se facilitan si se modela la jerarquía de generalización con subentidades (alternativa 2). En este caso las operaciones hacen uso de los atributos heredados que ocurren en E_1 o en E_2 , pero el número de casos a los que se requiere acceso se limita a la población pertinente de casos (esto es, E_1 o bien E_2).

Entonces, se escoge la solución que sea favorable al más amplio volumen de operaciones, con base en la tabla de volumen de acceso de operaciones. Si el conjunto 2 predomina, no se requiere una elección posterior: se escoge la alternativa 2. Si el conjunto 1 predomina, se tendrá que elegir después entre la alternativa 1 y la 3, con base en un análisis posterior de las operaciones, sobre todo la actualización, como sigue:

1. Las operaciones que usan atributos tanto de la superentidad como de la sub-entidad al mismo tiempo se facilitan con la alternativa 1.
2. Las operaciones que usan atributos de la superentidad o bien de la sub-entidad (pero nunca de las dos a la vez) se facilitan con la alternativa 3.

Al tomar esta decisión se debe tener presente que la alternativa 3 es más compleja y requiere más trabajo en el caso de actualizaciones.

En el caso de jerarquías de generalización de varios niveles se puede aplicar el procedimiento anterior recursivamente, empezando por la base de la jerarquía y eliminando una generalización por vez hasta llegar a la entidad raíz.

11.5. Partición de entidades

La razón para dividir las entidades es reorganizar la distribución de los casos (partición horizontal) o de los atributos (partición vertical) de manera que una entidad incluya atributos o casos a los que las operaciones requieran acceso simultáneo con frecuencia. Como en las actividades previas, las operaciones desempeñan un papel importante en esta decisión. Otra razón es la seguridad, para lo cual se otorga a ciertos individuos acceso selectivo a ciertas entidades.

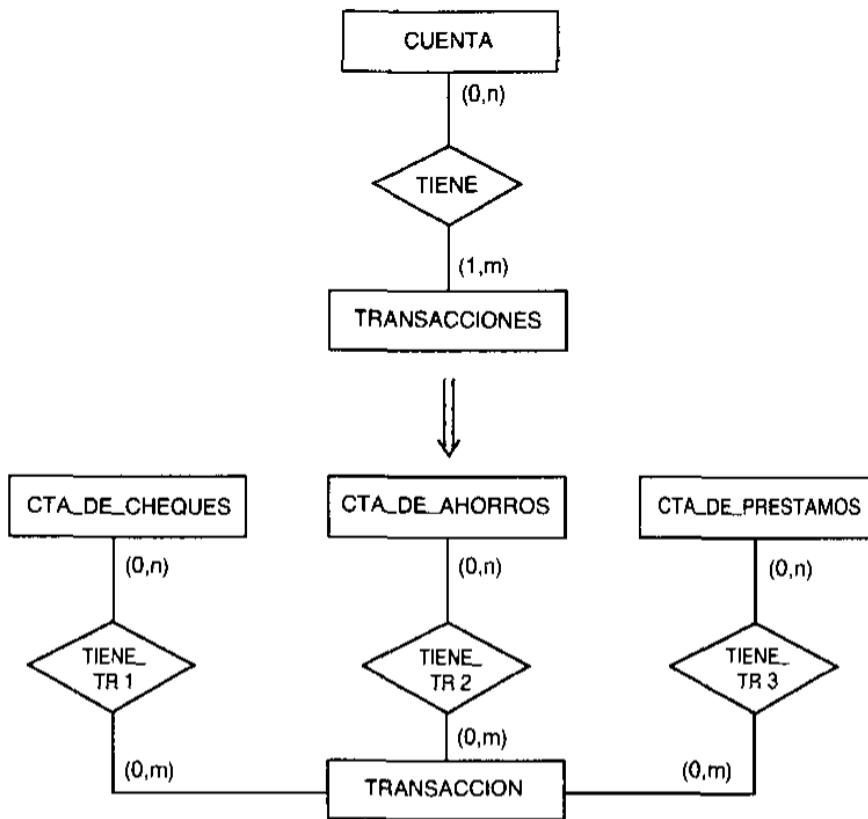
La partición de entidades se realiza dividiendo una entidad E en dos o más entidades, es decir, E_1, E_2, \dots, E_n . Hay dos clases de partición de entidades:

1. **Partición horizontal:** Aquí la partición se refiere a los casos de E: cada una de las entidades E_1, \dots, E_n tiene un grupo discreto de casos y tiene todos los atributos de E. Cada una satisface alguna afirmación (condición) que la distingue de las otras entidades.
2. **Partición vertical:** Aquí la partición se refiere a los atributos de E: cada una de la entidades E_1, \dots, E_n tiene exactamente los mismos casos de E pero tiene su propio grupo de atributos. Todas las entidades E_1, \dots, E_n deben tener identificadores. Las entidades $E_i, E_{i+1}, (i = 2, 3, \dots, n)$ están relacionadas por interrelaciones uno a uno.

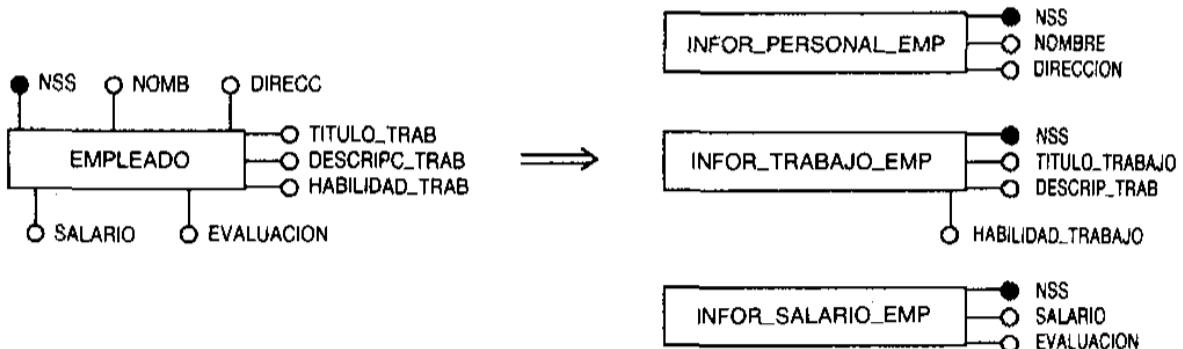
Las particiones de entidades pueden superponerse. En particiones horizontales superpuestas el mismo *caso* puede pertenecer a varias particiones, en particiones verticales superpuestas el mismo *atributo* pertenece a varias particiones.

La figura 11.7 muestra la entidad CUENTA, que se divide horizontalmente en las entidades CUENTA_CHEQUES, CUENTA_AHORRO y CUENTA_PRESTAMO. Esta distinción se basa en el tipo de cuenta. La entidad EMPLEADO se divide verticalmente en las entidades INFOR_PERSONAL_EMP, INFOR_TRABAGO_EMP e INFOR_SALARIO_EMP; la primera contiene todos los atributos relativos a la información personal de un empleado, y las otras contienen la información relacionada con el trabajo de un empleado.

La figura 11.8 ofrece un ejemplo general de división horizontal y vertical para



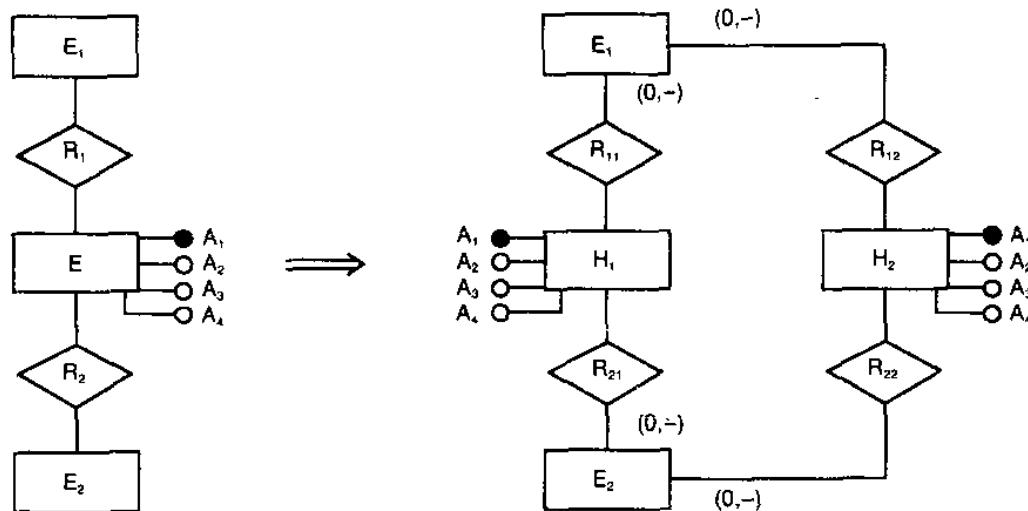
(a) Partición horizontal



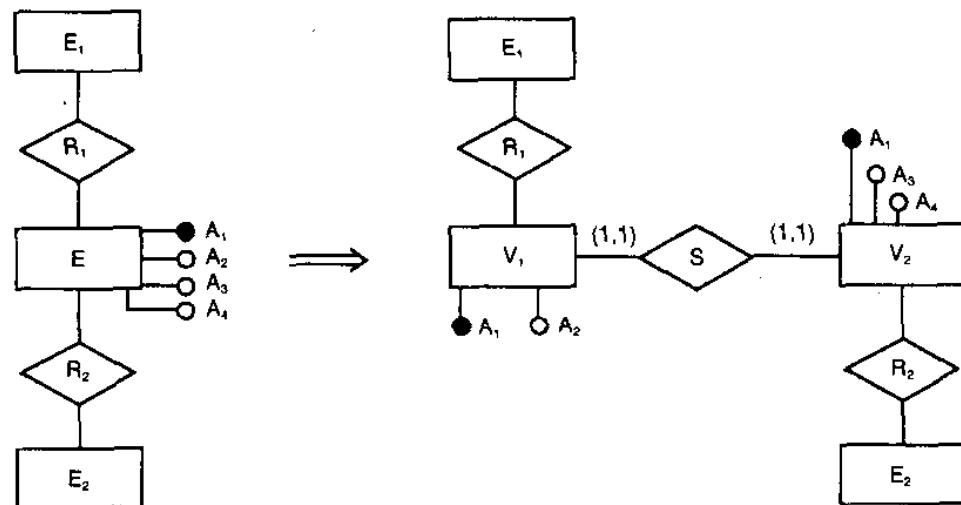
(b) Partición vertical

Figura 11.7. Partición horizontal y vertical.

ilustrar sus implicaciones. En la figura 11.8a la entidad E está dividida horizontalmente en H_1 y H_2 ; la interrelación R_1 está también dividida en R_{11} y R_{12} . De igual forma, la interrelación R_2 está dividida en R_{21} y R_{22} . Obsérvese que los valores de cardinalidad mínima de E_1 y E_2 en las interrelaciones R_{11} , R_{12} , R_{21} ,



(a) Partición horizontal



(b) Partición vertical

Figura 11.8. Casos generales de partición horizontal y vertical.

y R₂₂ son cero. Los valores de cardinalidad máxima se muestran como guiones. Si la división horizontal no es superpuesta, el valor en lugar del guión sería 1. Si la división es superpuesta, el valor puede ser mayor que 1 y menor o igual que el número de fragmentos horizontales. En la figura 11.8b la entidad E está dividida verticalmente en V₁ y V₂, la interrelación S que une V₁ y V₂ es una interrelación uno a uno con cardinalidad de (1, 1) en cualquier dirección. Esta interrelación permite «recuperar» los datos pertenecientes a la entidad original.

Las particiones de entidades pueden ser superpuestas. En las particiones horizontales superpuestas, el mismo *caso* puede pertenecer a varias particiones; en

las particiones verticales superpuestas, el mismo *atributo* pertenece a varias particiones.

Consideremos los dos ejemplos de aplicación de partición horizontal y vertical que se aprecian en la figura 11.7. Si suponemos que el procesamiento de las cuentas de cheques, ahorros y préstamos ocurre en diferentes aplicaciones, es razonable dividirlas de manera que se pueda reducir el acceso total a la base de datos. Esta distinción es más importante en el procesamiento por lotes, que puede producir un libro mayor completo de un tipo de cuenta dado. En el caso de operaciones en línea que implican acceso aleatorio a una cuenta por su número de cuenta, la partición podría no tener un efecto significativo. Como ejemplo adicional, se puede pensar que la entidad TRANSACCIONES se divide en TRANSACCIONES_MAYORES y TRANSACCIONES_MENORES. Esto puede facilitar la aplicación de diferentes políticas, como por ejemplo los procedimientos de autorización a estos dos tipos de transacciones (véase el ejercicio 11.6). La figura 11.7b muestra la división vertical de empleado en tres entidades: INFOR_PERSONAL_EMP, INFOR_TRABAJO_EMP e INFOR_SALARIO_EMP. Esta separación puede surgir porque la mayoría de las aplicaciones requieren acceso a los datos básicos personales de cada empleado que se encuentran en INFOR_PERSONA_EMP. La entidad INFOR_TRABAJO_EMP tiene atributos que sólo requieren las aplicaciones que necesitan saber qué personas trabajan en qué proyectos o que modifican las descripciones de trabajo. INFOR_SALARIO_EMP, por otro lado, contiene información confidencial sobre salarios y evaluaciones del desempeño.

Las particiones horizontal y vertical son útiles sobre todo en el diseño de bases de datos distribuidas, donde se trata de mantener los datos pertinentes (ambos, casos y atributos) disponibles localmente hasta donde sea posible. Esto requiere una partición vertical y horizontal de la entidad global, y la asignación posterior de particiones individuales a las bases de datos locales.

11.5.1. Partición de interrelaciones

Conceptualmente, un posible efecto adverso de la división es la proliferación de interrelaciones (véase, por ejemplo, el ejercicio 11.6). Las interrelaciones se dividen automáticamente cada vez que se dividen las entidades participantes.

En algunas situaciones de interrelaciones de uno a muchos y de muchos a muchos, la interrelación misma puede ser dividida horizontalmente por alguna razón de peso; se muestra un ejemplo en la figura 11.9. La interrelación SE_INSCRIBE es una interrelación de muchos a muchos entre ESTUDIANTE y CURSO. Con base en el atributo SEMESTRE de SE_INSCRIBE, esta interrelación puede dividirse en INSCRIPCION_OTOÑO90, INSCRIPCION_INVIERNO91 y así sucesivamente. Por lo regular, se obtendría acceso a los datos de inscripción semestre por semestre; así, el acceso a estas interrelaciones puede ser mucho más eficiente.

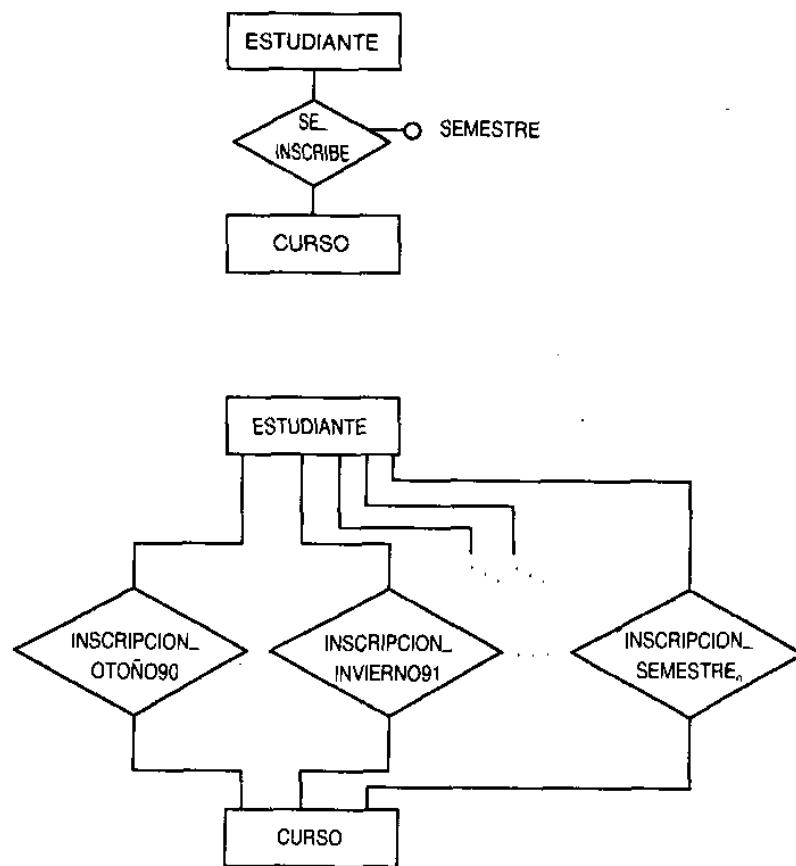


Figura 11.9. División de la interrelación SE_INSCRIBE.

11.6. Fusión de entidades e interrelaciones

Fusionar entidades e interrelaciones es lo opuesto a dividirlas. La fusión se aplica cuando dos o más entidades se usan juntas para muchas operaciones y consiste en integrarlas en una sola entidad. Todas las interrelaciones que las enlazan se fusionan en forma correspondiente. La fusión de entidades es un paso importante del diseño de bases de datos complejas con operaciones que navegan por múltiples entidades e interrelaciones. Fusionar algunas entidades y las interrelaciones que las enlazan lleva a un agrupamiento de información así que permite accesos eficientes para las aplicaciones que necesitan esos cúmulos de información.

Un efecto adverso de la fusión propuesta es que en algunos casos resulta equivalente a la «desnormalización»: el esquema conceptual normalizado se transforma en uno menos normalizado, lo que implica la adición de un grupo de anomalías, como las de actualización, que requieren trabajo adicional. Por consiguiente, los pros y contras de la fusión deben evaluarse cuidadosamente.

No se afecta el nivel de normalización si la interrelación entre las entidades

es de uno a uno. Si la interrelación es de uno a muchos, la entidad fusionada puede violar la tercera forma normal (analizada en el capítulo 6); si la interrelación es de muchos a muchos, puede violar la segunda forma normal. La figura 11.10 ilustra la fusión de entidades conectadas por diferentes tipos de interrelaciones. La figura 11.10a muestra una fusión de entidades donde PEDIDO y FACTURA_EMBARQUE están relacionadas por una interrelación uno a uno. Esta unión no genera violación de la normalización. La figura 11.10b muestra la unión de entidades con una interrelación intermedia de uno a muchos. En este caso, la dependencia funcional introduce una anomalía después de la fusión, lo que puede llevar a una violación de la tercera forma normal. La figura 11.10c muestra la fusión de entidades con una interrelación intermedia de muchos a muchos. En este caso, las dependencias

$$\begin{aligned} \text{NM_PROY} &\rightarrow \text{NOMBRE_DIR}, \text{ y} \\ \text{NM_DEPT} &\rightarrow \text{NUM_DE_EMPLEADOS} \end{aligned}$$

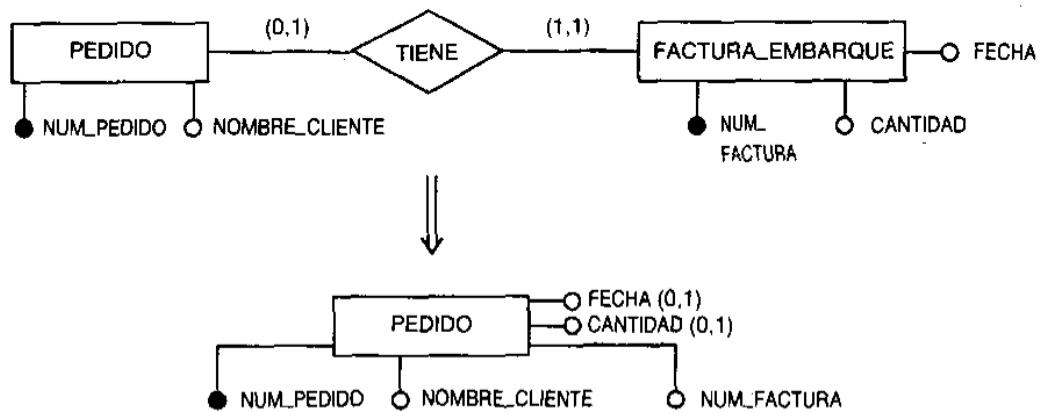
dentro de la entidad resultante introducen una violación a la segunda forma normal.

Los tres ejemplos de la figura 11.10 simplifican la estructura de la base de datos y, por consiguiente, simplifican las operaciones que navegan a lo largo de las interrelaciones intermedias. Sin embargo, la actualización de los atributos en el lado derecho de las dependencias antes mencionadas resulta más costoso.

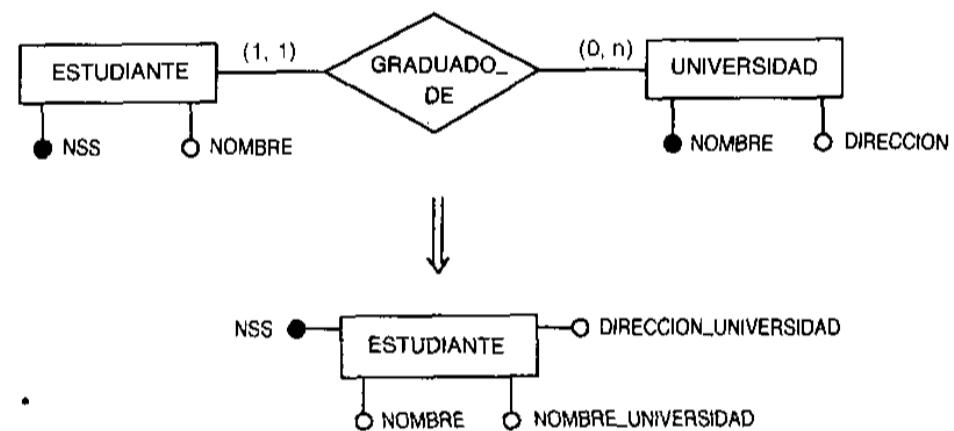
Puede resultar sorprendente que se hable de la división de entidades y la fusión de entidades como operaciones de simplificación, porque son opuestas. Obviamente, estas transformaciones no deben ser usadas una después de la otra con la misma entidad de manera que se anulen sus efectos. Realmente, estas operaciones se usan muy poco en la fase temprana del diseño lógico; es más probable que se usen en el contexto de distribuir una base de datos o de consolidar bases de datos existentes. Las bases de datos distribuidas se benefician del paralelismo que es posible al introducir la división horizontal. Cada fragmento, después de la división, se puede guardar independientemente, y las subconsultas pueden realizarse en paralelo.

11.7. Selección de claves primarias

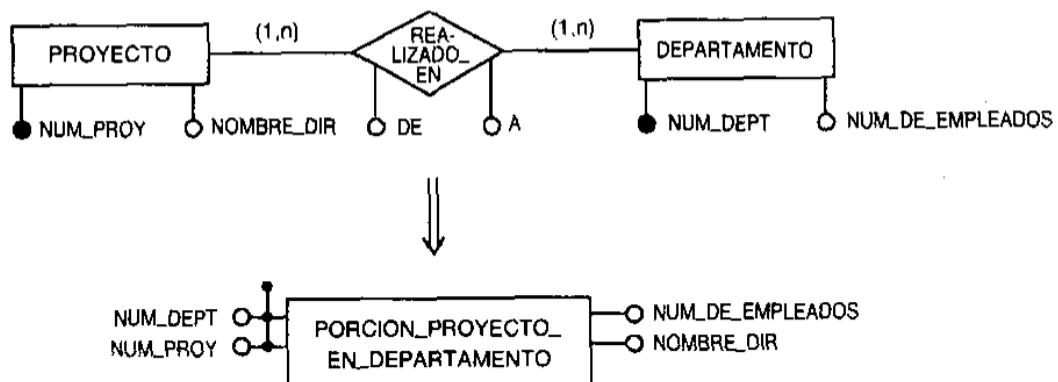
La mayoría de los DBMS requieren que se escoja como *clave primaria* uno de los identificadores de una entidad. Esta clave sirve como identificador para hallar un caso único de la entidad, dado el valor de la clave primaria. Algunos sistemas (por ejemplo, el DBMS relacional DB2) quizás no soliciten la clave primaria durante la definición de los datos, pero pueden aceptarla como una **clave única indexada** (o indexada). Por lo general, la clave primaria se asocia con una realización más rápida del acceso a la entidad en el diseño físico subsecuente. Por tanto, un criterio de decisión es seleccionar como clave primaria el identi-



(a) Fusión de entidades con una interrelación intermedia de uno a uno



(b) Fusión de entidades con una interrelación intermedia de uno a muchos



(c) Fusión de entidades con una interrelación intermedia de muchos a muchos

Figura 11.10. Casos de fusión de entidades e interrelaciones.

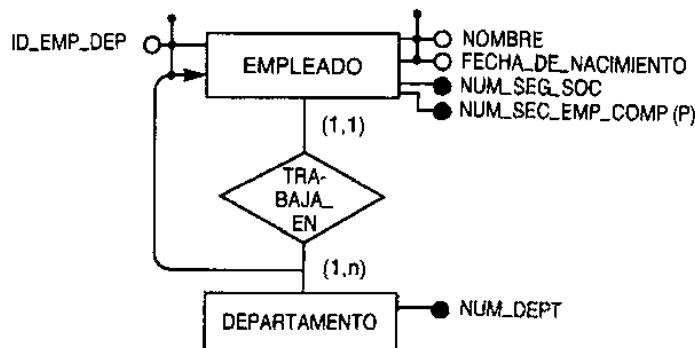


Figura 11.11. Selección de la clave primaria para la entidad EMPLEADO.

ficador usado para accesos directos por el máximo número de operaciones. Si una entidad tiene múltiples identificadores, se debe designar uno de ellos como clave primaria de la entidad. Un segundo criterio de decisión es preferir los identificadores simples a los múltiples y los internos a los externos; de esta manera, las claves primarias de las entidades se pueden mantener mínimas en tamaño y simples en estructura.

La entidad EMPLEADO de la figura 11.11 tiene cuatro identificadores: 1) el par NOMBRE, FECHA_NACIMIENTO; 2) el número de seguro social, NUM_SEG_SOC; 3) un número secuencial dentro de la compañía, NUM_SEC_EMP_COMP; y 4) un identificador dentro de cada departamento, ID_EMP_DEP, junto con el identificador externo del DEPARTAMENTO en el que trabaja el empleado. Con base en el segundo criterio, se seleccionan los identificadores simples internos NUM_SEG_SOC y NUM_SEC_COMP. Para seleccionar posteriormente entre estos dos habrá que basarse en el número de operaciones que usan los atributos arriba indicados para acceso directo; seleccionamos, por ejemplo, NUM_SEC_EMP_COMP. Es muy común durante el diseño físico definir múltiples índices para las entidades a las que se obtiene acceso frecuente. Por esto, aunque NUM_SEC_EMP_COMP se considera como clave primaria, podríamos definir un índice único según NUM_SEG_SOC. La clave primaria se marca en el esquema con (P).

11.8. Diseño lógico de alto nivel en el caso de estudio

Ahora se aplicará la metodología explicada anteriormente para el caso de estudio del capítulo 10. Las partes del esquema del caso de estudio que se desea analizar más, y su correspondiente información de volumen de datos, se muestran en la figura 11.12. La tabla 11.4 muestra la tabla completa de volumen de datos para entidades, interrelaciones y atributos. Sólo se considerarán las operaciones para las cuales se dieron los esquemas de navegación en el capítulo 10; éstas son sólo operaciones representativas. Por consiguiente, sólo se bosquejará la natu-

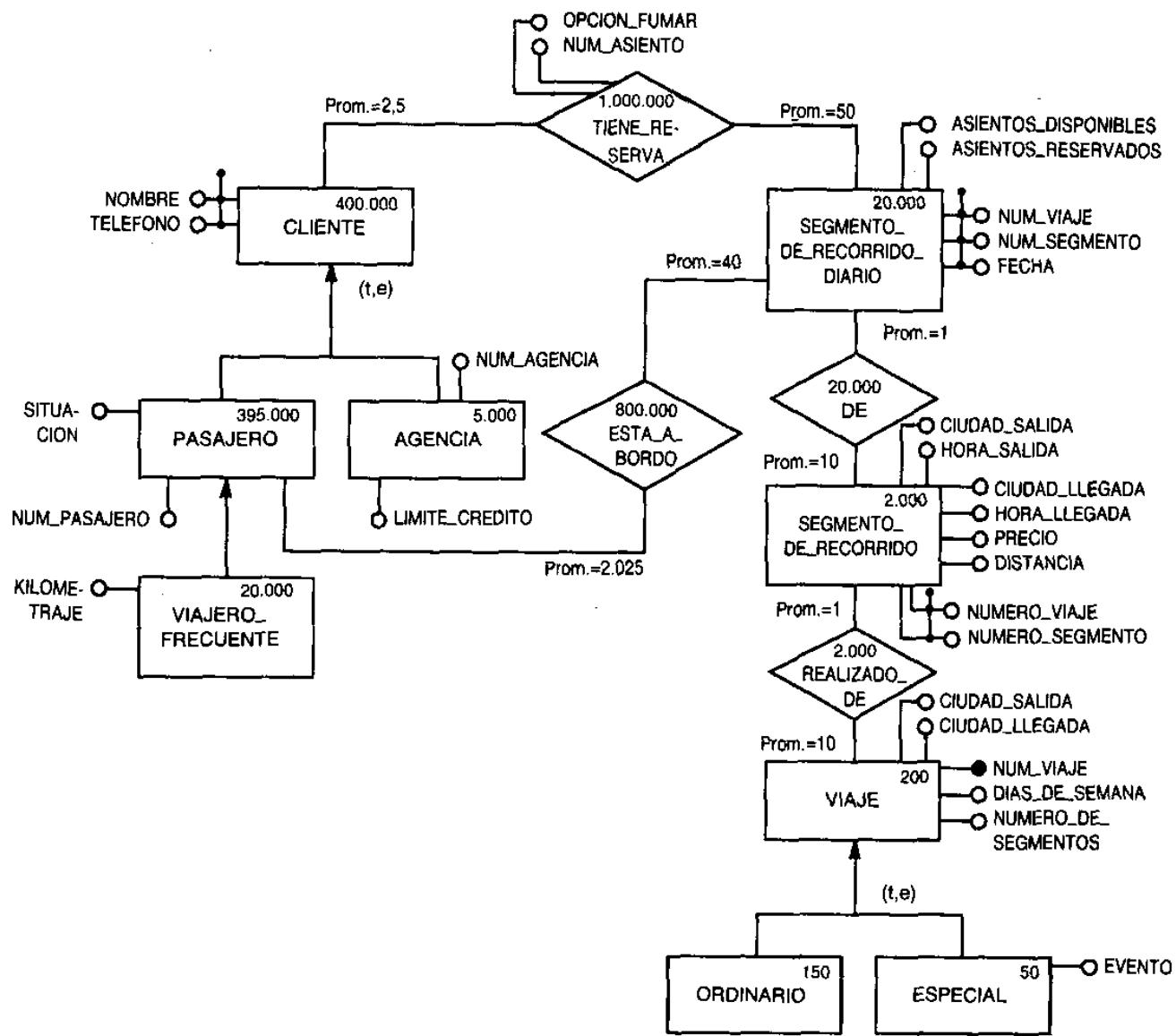


Figura 11.12. Parte pertinente del esquema de datos del caso de estudio con información de volumen de datos.

raza de la modificación del esquema que puede resultar apropiada para la base de datos del caso de estudio. La información de frecuencia de operaciones aparece en la tabla 11.5. No se da una tabla detallada de volumen de acceso de operaciones para el caso de estudio; esto se deja como ejercicio para el lector. Se hará el análisis sobre una base diaria.

11.8.1. Decisiones sobre los datos derivados

Para cada dato derivado, se debe considerar las operaciones sobre las que influye ese dato (la operación recupera los datos derivados o bien participa en su

Tabla 11.4. Tabla de volumen de datos para el esquema del caso de estudio

Concepto	Tipo	Volumen
CLIENTE	E	400.000
NOMBRE	A	400.000
TELEFONO	A	400.000
PASAJERO	E	395.000
NUM_PASAJERO	A	395.000
VIAJERO_FRECUENTE	E	20.000
KILOMETRAJE	A	20.000 ^a
AGENCIA	E	5.000
NUM_AGENCIA	A	5.000
LIMITE_CREDITO	A	5.000
SEGMENTO_DE_RECORRIDO_DIARIO	E	20.000
NUMERO_VIAJE	A	200
NUMERO_SEGMENTO	A	200
FECHA	A	90 ^c
ASIENTOS_DISPONIBLES	A	100 ^d
ASIENTOS_RESERVADOS	A	100 ^d
SEGMENTO_DE_RECORRIDO	E	2.000
NUMERO_VIAJE	A	200
NUMERO_SEGMENTO	A	200
CIUDAD_SALIDA	A	150
HORA_SALIDA	A	2.000 ^a
CIUDAD_LLEGADA	A	150
HORA_LLEGADA	A	2.000 ^a
PRECIO	A	2.000 ^a
DISTANCIA	A	2.000 ^a
VIAJE	E	200
NUMERO_VIAJE	A	200
NUMERO_DE_SEGMENTOS	A	10
CIUDAD_SALIDA	A	150
CIUDAD_LLEGADA	A	150
DIAS_DE_SEMANA	A	128 ^e
(VIAJE) ORDINARIO	E	150
(VIAJE) ESPECIAL	E	50
EVENTO	A	50
VIAJE_DIARIO	E	2.000
NUMERO_VIAJE	A	200
FECHA	A	90 ^c

Tabla 11.4. Tabla de volumen de datos para el esquema del caso de estudio (*continuación*)

Concepto	Tipo	Volumen
AUTOBUS	E	500
NUM_MATRICULA	A	500
ID_AUTOBUS	A	500
ASIENTOS	A	100 ^d
MARCA	A	10
ULTIMA_REVISION	A	365
PROBLEMA_AUTOBUS	E	1.500
CODIGO_PROBLEMA	A	20
DESCRIPCION	A	1.500
CONDUCTOR	E	300
ID_CONDUCTOR	A	300
NOMBRE	A	300 ^a
TELEFONO	A	300 ^a
DIRECCION	A	300 ^a
AUSENCIA_CONDUCTOR	E	3.000
ID_CONDUCTOR	A	300
FECHA	A	90
CAUSA	A	20 ^b
TIENE_RESERVA	R	1.000.000 ^c
NUM_ASIENTOS	A	100 ^d
OPCION_FUMAR	A	2
ESTA_A_BORDO	R	800.000
DE (de SEGMENTO_DE_RECORRIDO_DIARIO)	R	20.000
DE (de VIAJE_DIARIO)	R	2.000
REALIZADO_DE (de SEGMENTO_DE_RECORRIDO)	R	2.000
REALIZADO_DE (de SEGMENTO_DE_RECORRIDO_DIARIO)	R	20.000
USA	R	500
CON (de AUTOBUS)	R	1.500
CONDUCIDO_POR	R	2.000
CON (de CONDUCTOR)	R	3.000

Nota: Se supone que sólo una parte de todos los segmentos de recorrido diario y de los viajes diarios para viajes pasados y futuros se mantienen en línea. Sólo éstos se han incluido en la tabla.

^a Peor caso estimado.

^b Indica atributo polivalente.

^c Supone datos de SEGMENTO_DE_RECORRIDO_DIARIO sólo durante tres meses.

^d Supone que la capacidad máxima del autobús es 100

^e Todas las combinaciones posibles de días de la semana (2^7).

Tabla 11.5. Tabla de frecuencia de operación para el caso de estudio

Nombre/descripción de la operación	Frecuencia	Tipo (en línea/ por lotes)
O1 BUSCAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA	50 veces al día	EL
O2 RECUPERAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA y CIUDAD_LLEGADA	200 veces al día	EL
O3 RECUPERAR VIAJES CON PARADA INTERMEDIA EN UNA CIUDAD DADA	5 veces al día	EL
O4 CREAR UN REGISTRO DE CLIENTE NUEVO	500 veces al día	EL
O5 HACER UNA RESERVA (CLIENTE EXISTENTE)	20.000 veces al día	EL ^a
O6 BORRAR RESERVA DE UN VIAJE PASADO	70 veces al día	EL
O7 BORRAR UN REGISTRO DE CLIENTE	1 vez al día	EL
O8 CALIFICAR UN CLIENTE COMO VIAJERO FRECUENTE	10 veces al día	EL
O9 RECUPERAR LA CANTIDAD DE KILOMETROS ACUMULADOS POR LOS VIAJEROS FRECUENTES	Una vez al mes	PL
O10 ACTUALIZAR EL KILOMETRAJE DE UN VIAJERO FRECUENTE	Una vez al día	PL
O11 RECUPERAR TODAS LAS RESERVAS ACTUALES PARA UN VIAJE DETERMINADO EN UNA FECHA DETERMINADA	100 veces al día	EL ^b
O12 RECUPERAR TODAS LAS RESERVAS ACTUALES DE UNA AGENCIA DETERMINADA	10 veces al día	EL ^b

^a Se supone que un volumen muy alto de reservas es congruente con el hecho de que los volúmenes de datos reflejan datos guardados sólo durante 90 días.

^b Operación que tiene un informe fuera de línea.

cálculo). La tabla 11.6 muestra una matriz de operaciones y sus relaciones con los elementos de datos derivados. Primero se analiza ASIENTOS_RESERVADOS. En 20.000 operaciones O5: HACER UNA RESERVA, este dato se requiere para 20.000 SEGMENTO_DE_RECORRIDO_DIARIO. Si no está ahí, para las 20.000 ocurrencias de SEGMENTO_DE_RECORRIDO_DIARIO (algunas de ellas pueden ser las mismas), se tendrá que calcular obteniendo acceso a los PASAJEROS que tienen reserva para ese segmento. Si suponemos que la mitad del número de reservas normales existe en el momento de hacer una nueva reserva, el número será 50/

Tabla 11.6. Datos derivados visitados por las operaciones del caso de estudio

Descripción de la operación	Influencia sobre el dato derivado					
	ASIENTOS_RESERVEDOS	ASIENTOS_DISPONIBLES	NUMERO_DE_SEGMENTOS	CIUDAD_SALIDA	CIUDAD_LLEGADA	
O1 BUSCAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA	N	N	N	N	S	
O2 RECUPERAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA Y CIUDAD_LLEGADA	N	N	N	S	S	
O3 RECUPERAR VIAJES CON PARADA INTERMEDIA EN UNA CIUDAD DADA	N	N	N	S	S	
O4 CREAR UN REGISTRO DE CLIENTE NUEVO	N	N	N	N	N	
O5 HACER UNA RESERVA (CLIENTE EXISTENTE)	S	S	N	N	N	
O6 BORRAR RESERVA DE UN VIAJE PASADO	S	S	N	N	N	
O7 BORRAR UN REGISTRO DE CLIENTE	N	N	N	N	N	
O8 CALIFICAR UN CLIENTE COMO VIAJERO FRECUENTE	N	N	N	N	N	
O9 RECUPERAR LA CANTIDAD DE KILOMETROS ACUMULADOS POR LOS VIAJEROS FRECUENTES	N	N	N	N	N	
O10 ACTUALIZAR EL KILOMETRAJE DE UN VIAJERO FRECUENTE	N	N	N	N	N	
O11 RECUPERAR TODAS LAS RESERVAS ACTUALES PARA UN VIAJE DETERMINADO EN UNA FECHA DETERMINADA	N	N	N	N	N	
O12 RECUPERAR TODAS LAS RESERVAS ACTUALES DE UNA AGENCIA DETERMINADA	N	N	N	N	N	

2 = 25 reservas existentes. De manera que, en total, se necesitan $20.000 \times 25 = 500.000$ recuperaciones (de TIENE_RESERVA a CLIENTE). Para las 70 ocurrencias de O6: ELIMINAR_RESERVA_DE_UN_VIAJE_ANTERIOR, serán necesarias 70 actualizaciones del dato ASIENTOS_RESERVEDOS, si lo mantenemos. El almacenamiento adicional es, digamos, 3 bytes \times 20.000 casos de SEGMENTO_DE_RECORRIDO_DIARIO. Así, en conclusión, mantener este dato nos ahorra 500.000 –

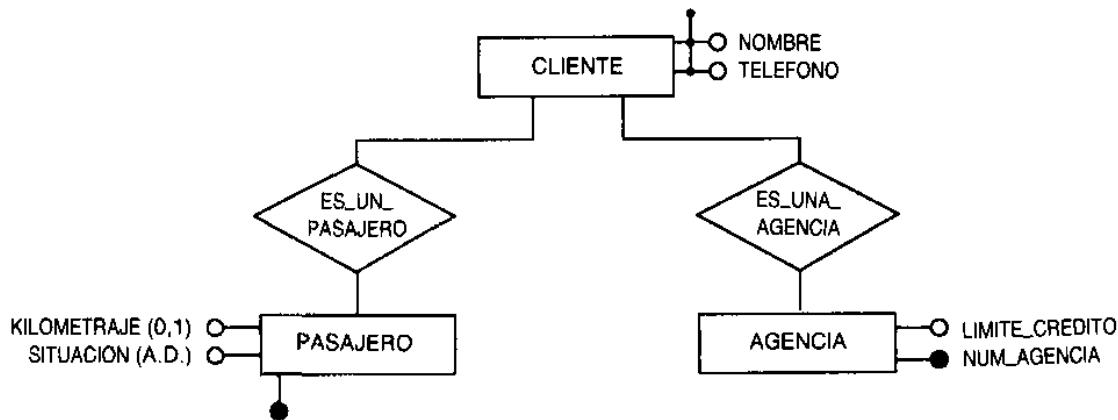


Figura 11.13. Modelado de la jerarquía de generalización de CLIENTE.

$20.000 = 480.000$ accesos, y nos cuesta un almacenamiento y un tiempo de procesamiento (por actualización) adicionales insignificantes, como se describió antes. El mismo análisis se aplica a ASIENTOS_DISPONIBLES. Por eso se mantienen estos dos atributos derivados en el esquema.

Acerca de NUMERO_DE_SEGMENTOS en VIAJE, ninguna de las operaciones hasta ahora mencionadas es influida por él, de manera que se decide retirar este atributo del esquema. Finalmente, tanto CIUDAD_LLEGADA como CIUDAD_SALIDA son consultados por las tres primeras operaciones significativas; implican un costo de almacenamiento, pero no hay costo de tiempo de ejecución por actualización, ya que son valores asignados automáticamente (por el sistema) cuando se crean a partir de los SEGMENTO_DE_RECORRIDO de ese VIAJE, de manera que se decide mantenerlos en la entidad SEGMENTO_DE_RECORRIDO aunque sean constantes.

11.8.2. Integración de las jerarquías de generalización

En relación a la generalización entre CLIENTE, PASAJERO, y AGENCIA, se observa que la operación más importante que atañe a esta generalización es O5: HACER UNA RESERVA. Ya que el operador sabe si el cliente es un pasajero o una agencia, naturalmente podría tener dos versiones de la operación O5: HACER UNA RESERVA. Por eso, es mejor separar las dos entidades, ya que todas las demás operaciones implican sólo a PASAJERO. La generalización se modela con la alternativa 3 (apartado 11.4.3) usando interrelaciones. El resultado se muestra en la figura 11.13. A la subentidad VIAJERO_FRECUENTE se obtiene acceso por separado de la superentidad PASAJERO sólo en las operaciones O8, O9, y O10, que son muy poco frecuentes. Por añadidura, el único atributo adicional de la subentidad es KILOMETRAJE. Por eso, la entidad VIAJERO_FRECUENTE puede integrarse en PASAJERO. El atributo discriminativo situación se añade a la entidad PASAJERO para indicar si un caso dado de PASAJERO perte-

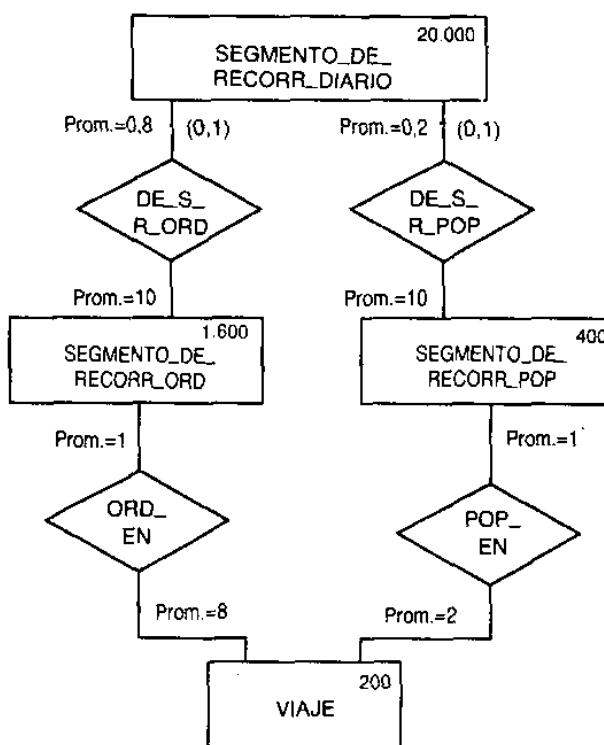


Figura 11.14. División de la entidad **SEGMENTO_DE_RECORRIDO** en **SEGMENTO_DE_RECORRIDO_ORD** y **SEGMENTO_DE_RECORRIDO_POP**.

nece o no a la subentidad **VIAJERO_FRECUENTE**. Respecto a la segunda generalización, que implica **VIAJE** y **ORDINARIO** y **ESPECIAL**, sólo el atributo **EVENTO** distingue las dos entidades. Se puede, pues, integrar las tres entidades en la única entidad **VIAJE** (y añadir a **VIAJE** el atributo discriminativo **TIPO** así como el atributo **EVENTO**).

11.8.3. Partición de entidades

En este caso, un análisis de las operaciones no proporciona bases inmediatas para una partición de las entidades. Después de una observación más detenida, supongamos, como sería razonable en muchas situaciones, que un número limitado de segmentos de recorrido (digamos, 10%) participan en la mayoría de las ejecuciones de las operaciones O1, O2 y O3 (digamos, 90% por volumen total). En este caso, nos conviene dividir la entidad **SEGMENTO_DE_RECORRIDO** en dos entidades denominadas **SEGMENTO_DE_RECORRIDO_ORD** y **SEGMENTO_DE_RECORRIDO_POP** para segmentos de recorridos ordinarios y populares. En consecuencia, se debe duplicar las interrelaciones que conectan la entidad antigua **SEGMENTO_DE_RECORRIDO_DIARIO** y asignarlas a las dos entidades (Fig. 11.14). La figura supone que el 20% de todos los segmentos de recorrido (400 de ellos)

caen en la categoría popular. Obsérvese que la división de SEGMENTO_DE_RECORRIDO en dos subentidades puede llevar a la división correspondiente de la entidad SEGMENTO_DE_RECORRIDO_DIARIO en dos subentidades. Esto a su vez puede llevar a dividir la interrelación TIENE_RESERVA. Sin embargo, se decide *no* dividir SEGMENTO_DE_RECORRIDO_DIARIO ni TIENE_RESERVA para mantener el esquema manejable como caso de estudio⁶.

11.8.4. Agregación de entidades e interrelaciones

La tabla 11.7 muestra la correspondencia entre las entidades y operaciones (para simplificar se usan las entidades CLIENTE, SEGMENTO_DE_RECORRIDO_DIARIO, SEGMENTO_DE_RECORRIDO y VIAJE). Dos pares de entidades que pueden integrarse son (CLIENTE, SEGMENTO_DE_RECORRIDO_DIARIO) y (SEGMENTO_DE_RECORRIDO, VIAJE). Al primer par tiene acceso comúnmente la operación O5, a través de la interrelación TIENE_RESERVA; al segundo, las operaciones O1, O2 y O3.

Considérese el par CLIENTE, SEGMENTO_DE_RECORRIDO_DIARIO en el caso en que CLIENTE es PASAJERO. Como CLIENTE tiene dos conexiones con SEGMENTO_DE_RECORRIDO_DIARIO, a saber, TIENE_RESERVA y ESTA_A_BORDO, podemos optar por cambiar a PASAJERO el identificador de SEGMENTO_DE_RECORRIDO_DIARIO de manera que se integre la interrelación TIENE_RESERVA y se mantenga la entidad PASAJERO y la interrelación ESTA_A_BORDO. Si se fusionan las dos entidades, se ahorra un acceso a la interrelación TIENE_RESERVA cada vez que se tiene acceso las dos entidades PASAJERO y SEGMENTO_DE_RECORRIDO_DIARIO.

Por ello, hay que calcular la redundancia que surge. Se tendrá un caso diferente de PASAJERO que corresponda a cada SEGMENTO_DE_RECORRIDO_DIARIO que él o ella reserve. Dado que, en promedio, un cliente tiene 2,5 reservas, el número de ocurrencias de PASAJERO se multiplicará 2,5 veces ($395.000 \times 2,5$). Cada vez que se realiza la operación O5 (lo que sucede 2.000 veces en un día), se debe crear una ocurrencia de PASAJERO, aunque ya exista el mismo pasajero. El número de actualizaciones de atributos como KILOMETRAJE y SITUACION en PASAJERO se multiplica también por un coeficiente de 2,5. Esto difícilmente es tolerable. Además, los atributos identificadores de SEGMENTO_DE_RECORRIDO_DIARIO se duplican, en promedio, un número de veces proporcional al número de pasajeros (esto es, 50) para cada SEGMENTO_DE_RECORRIDO_DIARIO. Esto es demasiado, por lo que se decide mantener las dos entidades PASAJERO y SEGMENTO_DE_RECORRIDO_DIARIO separadas.

Fusionar SEGMENTO_DE_RECORRIDO y VIAJE tiene también el efecto de multiplicar la información sobre VIAJE 10 veces (el número promedio de SEGMENTO_DE_RECORRIDO por VIAJE), de 200 a 2.000 ocurrencias. Esto resulta en

⁶ El lector atento quizás haya notado que no había razones de peso para la partición de entidades con base en la descripción original del caso de estudio. La hemos incorporado para fines de ilustración, pero no quisimos realizar una modificación amplia en el esquema lógico.

Tabla 11.7. Correspondencia entre entidades y operaciones para el caso de estudio

Descripción de la operación	Entidades implicadas			
	CLIENTE	SEGMENTO_DE_RECORRIDO	SEGMENTO_DE_RECORRIDO_DIARIO	VIAJE
O1 BUSCAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA	N	N	Y	Y
O2 RECUPERAR SEGMENTO_DE_RECORRIDO POR CIUDAD_SALIDA Y CIUDAD_LLEGADA	N	N	Y	Y
O3 RECUPERAR VIAJES CON PARADA INTERMEDIA EN UNA CIUDAD DADA	N	N	Y	Y
O4 CREAR UN REGISTRO DE CLIENTE NUEVO	Y	Y	N	N
O5 HACER UNA RESERVA (CLIENTE EXISTENTE)	Y	Y	N	N
O6 BORRAR RESERVA DE UN VIAJE PASADO	N	Y	N	N
O7 BORRAR UN REGISTRO DE CLIENTE	Y	N	N	N
O8 CALIFICAR UN CLIENTE COMO VIAJERO FRECUENTE	Y	N	N	N
O9 RECUPERAR LA CANTIDAD DE KILOMETROS ACUMULADOS POR LOS VIAJEROS FRECUENTES	Y	N	N	N
O10 ACTUALIZAR EL KILOMETRAJE DE UN VIAJERO FRECUENTE	Y	N	N	N
O11 RECUPERAR TODAS LAS RESERVAS ACTUALES PARA UN VIAJE DETERMINADO EN UNA FECHA DETERMINADA	N	Y	N	N
O12 RECUPERAR TODAS LAS RESERVAS ACTUALES DE UNA AGENCIA DETERMINADA	N	Y	N	N

una redundancia de datos porque todos los elementos de VIAJE, excepto NUMERO_VIAJE que es necesario para identificación externa, serán duplicados innecesariamente. También tiene asociados problemas de actualización cada vez

que una definición de VIAJE cambia. El otro efecto de la fusión es que la interrelación DE, que normalmente conecta VIAJE_DIARIO con VIAJE, también ascenderá 10 veces en su número de ocurrencias para conectar cada SEGMENTO_DE_RECORRIDO dentro del viaje diario. El beneficio es que las 255 recuperaciones por día (para O1, O2 y O3) no necesitarán tener acceso cada vez a una ocurrencia de VIAJE. Este beneficio no parece pesar más que la redundancia de almacenamiento y el aumento en 10 veces de las recuperaciones de la interrelación DE. Por eso se decide otra vez en contra de la fusión y se mantienen VIAJE y SEGMENTO_DE_RECORRIDO separados.

11.8.5. Selección de la clave primaria

Cada entidad, como aparece ahora, tiene exactamente un identificador, que se puede designar como clave primaria. El esquema conceptual a lógico final aparece en la figura 11.15. La parte del esquema enmarcada con la línea punteada no se ve afectada por las transacciones importantes que se describen en este capítulo, por lo que no se tendrá en cuenta esa parte en las subsecuentes etapas del diseño lógico que se describen en los próximos capítulos.

11.9. Resumen

En este capítulo analizamos el proceso global del diseño lógico, que implica dos fases. La primera, denominada diseño lógico *independiente del modelo*, se trató en este capítulo. Se mostró cómo caracterizar la carga de la base de datos en términos de información de volumen y frecuencia. Luego se comentó cómo usar esa información para simplificar el esquema ER dado. Las decisiones que consideramos están relacionadas con los atributos derivados, las jerarquías de generalización, la fusión y división de entidades e interrelaciones y, finalmente, la selección de claves primarias. Se trató el caso de estudio en el contexto de nuestro conocimiento de las operaciones y se simplificó hasta cierto punto.

En los próximos tres capítulos se considerará el diseño lógico de bases de datos para los tres modelos de datos más prominentes.

Ejercicios

- 11.1. Considere la ecuación (1) del apartado 11.2, en la que están implicadas las cardinalidades de dos entidades relacionadas y sus cardinalidades promedio respecto a la interrelación entre ellas. Construya un ejemplo real de una interrelación binaria, asignando valores significativos a la información de volumen de datos mencionada y verifique esa ecuación.

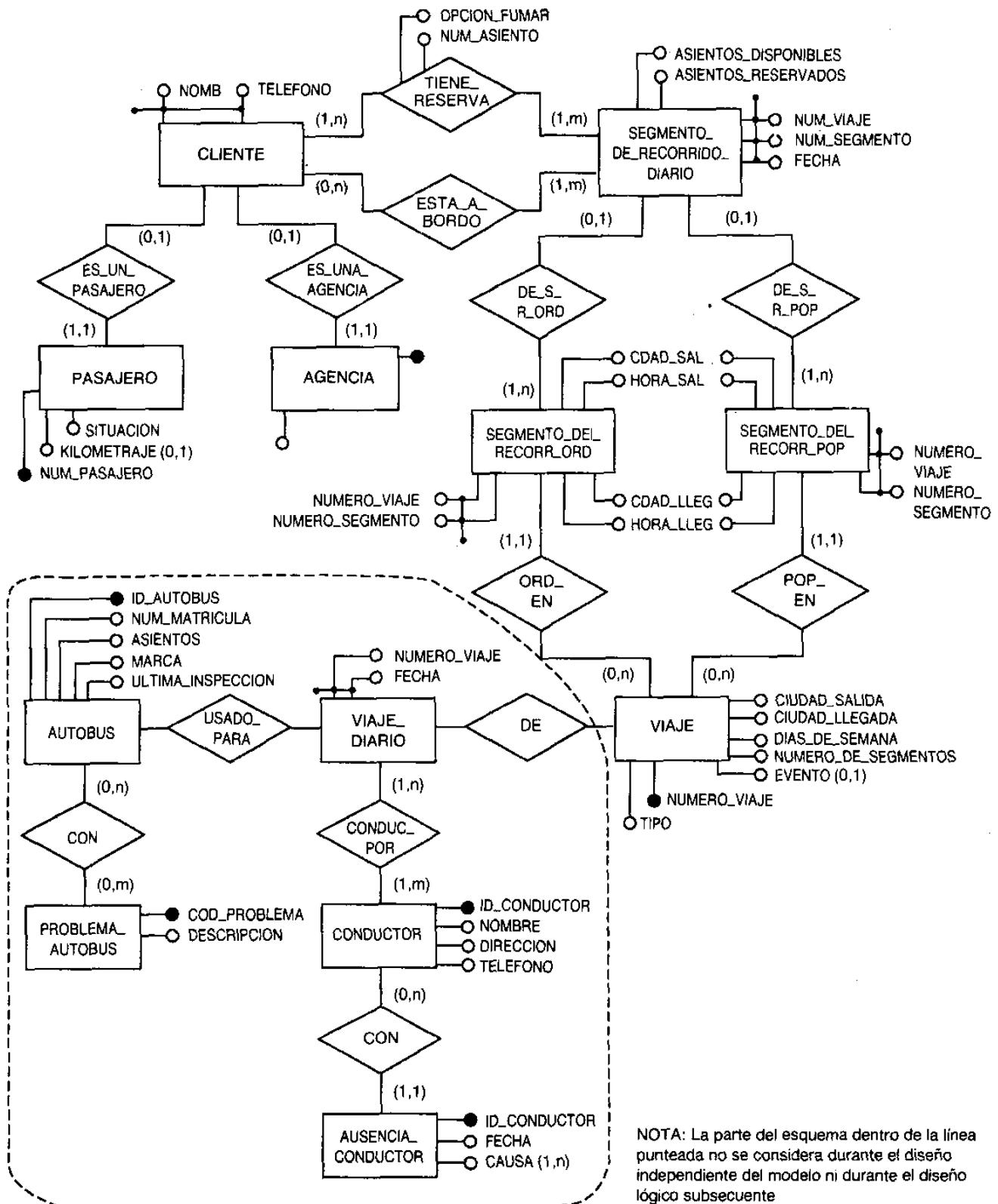


Figura 11.15. Esquema conceptual a lógico final.

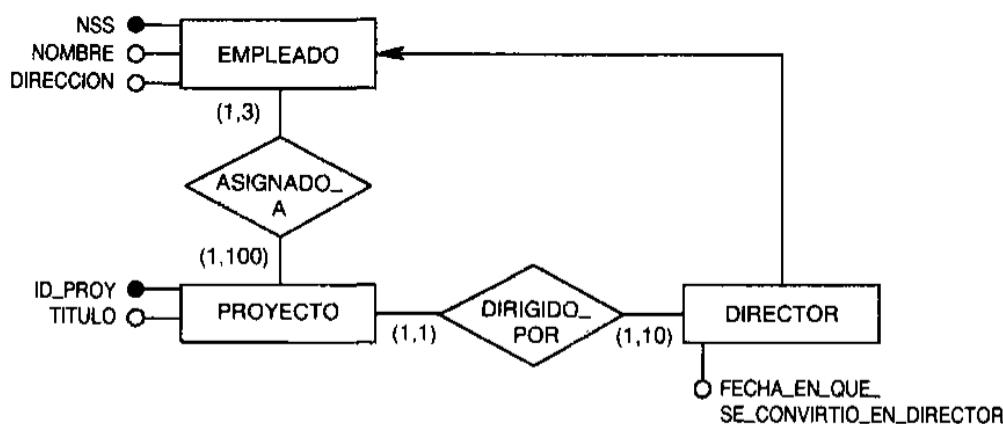


Figura 11.16. Un esquema de base de datos de proyectos.

- 11.2. Considere la tabla 11.3. Usando su conocimiento de las operaciones respecto al esquema en la figura 11.3, dibuje los esquemas de navegación para todas las operaciones. Aproveche la información de frecuencia de operaciones de la tabla 11.2 y los esquemas de navegación que haya dibujado para completar las tres últimas filas de esa tabla.
- 11.3. Considere los ejercicios 9.1 y 9.4 del capítulo 9. Asigne datos de carga realistas (volúmenes de datos, cardinalidades promedio de las interrelaciones) al esquema y suponga operaciones significativas con frecuencias de muestra apropiadas. Construya tablas de volumen de datos, frecuencia de operaciones y volumen de acceso de operaciones. Aplique entonces la metodología que se describió en este capítulo y convierta los esquemas dados en esquemas lógicos de ER. Justifique sus decisiones cuando sea posible.
- 11.4. Suponga que las siguientes operaciones están definidas en el esquema de la figura 11.16:
 - O1: CREAR UN NUEVO PROYECTO Y ASIGNAR EMPLEADOS AL PROYECTO.
 - O2: CAMBIAR EL DIRECTOR DEL PROYECTO.
 - O3: CAMBIAR LA ASIGNACION DE UN EMPLEADO DADO DE UN PROYECTO A OTRO.
 - O4: ENCONTRAR PROYECTOS QUE TENGAN ASIGNADOS MAS DE DIEZ EMPLEADOS.
 - O5: ENCONTRAR DIRECTORES QUE UTILICEN MAS DE UN PROYECTO.

La información de cardinalidad se proporciona en el esquema. Suponga volúmenes de datos y frecuencias de operaciones razonables para las operaciones mencionadas. En seguida, dibuje esquemas de navegación y establezca una tabla de volumen de acceso de operaciones para cada una de esas operaciones.

11.5. Diseñe el sistema de información de un laboratorio de diagnóstico médico. Son de interés varios tipos de personas: doctores, ayudantes y pacientes. Para cada uno de ellos se representa el apellido, la edad, y un código. Los pacientes (aproximadamente 60.000) necesitan reconocimientos médicos que deben reservarse con antelación. La historia de los reconocimientos de los últimos 180 días está almacenada en el sistema. Los reconocimientos son de distintos tipos, identificados por un código, una descripción y un precio. El precio del reconocimiento depende también del tipo de paciente. Cada doctor (hay 500 de ellos) y ayudante (hay 1.100 de ellos) está capacitado para realizar solamente ciertos tipos de reconocimientos. Los reconocimientos se hacen en salas específicas. Cada reconocimiento debe estar aprobado con la firma de un doctor.

Junto con los reconocimientos (200 por día), también se deben planificar las visitas (50 por día). Los reconocimientos pueden asignarse como continuación de visitas o de manera independiente. Cada reconocimiento tiene un resultado, y los resultados tanto de los reconocimientos como de las visitas deben almacenarse en una bitácora del paciente, que debe conservar la historia de las últimas 30 visitas o reconocimientos. Los doctores y ayudantes pueden hacer los reconocimientos, pero sólo los doctores hacen las visitas.

La principales operaciones de base de datos son:

- O1: CREAR UN PACIENTE NUEVO.
- O2: FIJAR UN RECONOCIMIENTO PARA EL PRIMER DIA DISPONIBLE.
- O3: IMPRIMIR LA HISTORIA CLINICA DE UN PACIENTE.
- O4: CALCULAR DATOS ESTADISTICOS RELATIVOS AL NUMERO DE PACIENTES VISITADOS POR CADA DOCTOR Y POR CADA ASISTENTE AL MES.
- O5: CAMBIAR UNA VISITA PROGRAMADA.
- O6: CAMBIAR LA CITA DE UN PACIENTE DE UN DOCTOR A OTRO.
- O7: CAMBIAR LOS PRECIOS DE LOS RECONOCIMIENTOS.
- O8: CALCULAR LA CANTIDAD TOTAL QUE DEBE PAGAR UN PACIENTE.
- O9: PREPARAR UN RECIBO PARA UN PACIENTE.
- O10: CAMBIAR EL TIPO DE UN PACIENTE.

Complete las especificaciones con atributos razonables para las entidades y suministre datos de carga para los conceptos en el esquema. Añada nuevas operaciones razonables y asigneles frecuencias. Desarrolle las tres tablas requeridas. En seguida, tome las decisiones analizadas en este capítulo. (Quizá desee realizar un diseño lógico de la base de datos para cierto modelo consultando el capítulo apropiado.)

- 11.6. En la figura 11.7a CUENTA ha sido dividida horizontalmente. Suponga que transacciones hubiese también sido dividida en TRANSACCIONES_MAYORES y TRANSACCIONES_MENORES. ¿Qué impacto tiene esto sobre la interrelación original TIENE? Dibuje un diagrama de ER que muestre las entidades divididas para CLIENTE y TRANSACCIONES con las interrelaciones apropiadas.

Bibliografía

- P. Bertaina, A. di Leva, y P. Giolito, «Logical Design in CODASYL and Relational Environment». En S. Ceri, ed., *Methodology and Tools for Data Base Design*, North-Holland, 1983.
- H. Sakai, «Entity Relationship Approach to Logical Database Design». En C. Davis, S. Jajodia, P. Ng, y R. Yeh, eds., *Entity-Relationship Approach to Software Engineering*, North-Holland, 1983.
- T. Teorey, J. Fry, «The Logical Record Access Approach to Database Design». *ACM Computing Surveys*, 12, núm. 2, 1980.
Estos artículos presentan tres versiones diferentes (pero similares) del método de recuento de visitas, presentado en este capítulo, para realizar el diseño lógico. Bertaina et al. usan una versión del modelo ER; Teorey y Fry usan una versión del modelo de redes.
- M. Bert, C. Ciardo, B. Demo, A. di Leva, F. Giolito, C. Iacobeli y V. Marrone, «The Logical Design in the DATAID Project: The Easymap System». En A. Albano, V. de Antonellis y A. di Leva, eds., *Computer-Aided Database Design: The DATAID Project*, Elsevier Science (North-Holland), 1985.
Este artículo describe una herramienta que ayuda al usuario a evaluar las frecuencias de las operaciones; el enfoque es similar al utilizado en este capítulo.
- M. Schkolnick y P. Sorensen, «Denormalization: Performance-Oriented DataBase Design Technique», *Proc. AICA Congress*, Bologna, Italia, 1980.
Este artículo muestra un ejemplo de metodología basada en la desnormalización, esto es, en la aplicación de transformaciones, tales como la agregación de entidades e interrelaciones, que hacen concesiones en cuanto a la normalización pero que mejoran el rendimiento.
- C. R. Carlson, W. Ji, A. K. Arora, «The Nested Entity-Relationship Model: A Pragmatic Approach to ER Comprehension and Design Layout». En F. Lochovsky, ed., *Prod. Eighth International Conference on Entity-Relationship Approach*, Toronto, North-Holland, 1989.
- T. Teorey, G. Wei, D. L. Bolton y J. A. Koenig, «ER Model Clustering as an Aid for User Communication and Documentation in Database Design», *Communications of the ACM*, 32, núm. 8, agosto de 1989.
Estos artículos presentan un enfoque de la agregación de un esquema completo ER en grupos. El proceso puede aplicarse recursivamente para llegar a altos niveles de abstracción y así mejorar la comunicación entre usuarios y diseñadores. El artículo de Carlson et al. intenta proporcionar una formalización de los conceptos relacionados con los diagramas ER anidados. Este tipo de modelado conceptual de múltiples niveles ayuda en las sucesivas fases de diseño.

S. Ceri, S. B. Navathe y G. Wiederhold, «Distribution Design of Logical Database Schemas», *IEEE Transactions on Software Engineering*, 9, núm. 4, julio dse 1983.

S. B. Navathe, S. Ceri, G. Wiederhold y J. Dou, «Vertical Partitioning Algorithms for Database Design», *ACM Transactions on Database Systems*, 9, núm. 4, diciembre de 1984.

Estos dos artículos analizan en detalle modelos de partición vertical u horizontal que resultan útiles en bases de datos distribuidas. Son parcialmente aplicables en el presente contexto.

Diseño lógico en el modelo relacional

Los DBMS relacionales son los sistemas de bases de datos más populares actualmente en mini y microcomputadores; se espera que lleguen a ser igual de populares en grandes computadores en los próximos años. Una lista de productos comerciales de DBMS relacionales en mini y macrocomputadores incluiría INGRES (ASK/Computer Systems, Inc.), SQL/DS y DB2 (IBM), ORACLE (ORACLE, Inc.), SYBASE (Sybase, Inc.) INFORMIX (Informix, Inc.), y UNIFY (Unify, Inc.). Esta lista no es de ningún modo exhaustiva. Estos DBMS relacionales están disponibles en una amplia gama de equipos y bajo varios sistemas operativos. En el apartado 12.1 se introduce brevemente el modelo relacional y se recomiendan al lector varios libros donde podrá encontrar otras descripciones del modelo, lenguajes y sistemas relacionales.

Este capítulo trata la transformación de un esquema lógico ER en un esquema relacional. Se espera que los diseñadores de bases de datos ejecutarán primero un diseño conceptual en un modelo de más alto nivel como el modelo de entidades-interrelaciones (ER) y harán corresponder los esquemas conceptuales con los esquemas relacionales. Por tanto, los objetivos de este capítulo van de acuerdo con los apremiantes requerimientos de hoy día. Suponemos que la reestructuración y simplificación que se mostraron en el capítulo 11 como parte del diseño lógico independiente del modelo ya han sido realizadas para convertir el esquema conceptual ER en un esquema ER conceptual a lógico.

El esquema ER lógico no es muy diferente de un esquema relacional: *no* incluye generalizaciones ni subconjuntos. Se necesita realizar otras simplificaciones: la eliminación de atributos compuestos y polivalentes, el modelado de identificadores externos como internos y la eliminación de interrelaciones. Este proceso de correspondencia se analiza en el apartado 12.2, que trata estos diferentes aspectos de manera independiente.

SQL es el lenguaje relacional más popular y se está convirtiendo en un estándar *de facto*. En el apartado 12.3, en lugar de dar una metodología de traducción formal y precisa, se muestra por medio de varios ejemplos cómo los esquemas de navegación pueden ser transformados en consultas de SQL. En el apartado 12.4 se completa el diseño relacional para el caso de estudio realizado en los

capítulos 10 y 11. El apartado 12.5 presenta un planteamiento general de retroingeniería de esquemas relacionales a esquemas ER. Esta «abstracción» de un esquema relacional de nivel más bajo a un esquema ER es útil para comprender los datos en las bases de datos ya existentes.

12.1. El modelo relacional

El modelo relacional fue propuesto en 1970 por Codd, y la popularidad de este modelo ha ido creciendo lenta pero firmemente, de manera que el término *relacional* ha llegado a ser común entre los profesionales informáticos. El modelo relacional de datos es un modelo simple, potente y formal para representar la realidad. También ofrece una base firme para enfocar y analizar formalmente muchos problemas relacionados con la gestión de bases de datos, como el diseño de la base de datos, la redundancia, la distribución, etcétera. El formalismo y una base matemática son las piedras angulares en el desarrollo de la teoría de las bases de datos relacionales.

Varios avances han tenido lugar en las últimas dos décadas que señalan la falta de expresividad y riqueza semántica del modelo relacional. Esto impulsó a Codd a publicar recientemente un libro titulado *The Relational Model for Database Management - Version 2* (véase la bibliografía). Sin embargo, la sencillez del modelo ha facilitado la construcción de lenguajes de consulta e interfaces para los usuarios finales de fácil utilización, y ha resultado en una productividad más alta de los programadores de bases de datos. La gestión de bases de datos relacionales será una tecnología muy útil durante varios años, o incluso décadas. En este capítulo se muestra cómo ir desde el modelo conceptual ER al modelo relacional para implantar una base de datos. También se muestra cómo tomar un conjunto de relaciones existentes y aplicarles *retroingeniería* para convertirlas en un esquema ER, captando la semántica propuesta para un mejor entendimiento.

El elemento básico del modelo es la **relación**, y un **esquema de base de datos relacional** es una colección de definiciones de relaciones. El esquema de cada relación es una agregación de **atributos**; el conjunto de todos los valores que puede adoptar un atributo en particular se denomina **dominio** de ese atributo.

Un **caso de relación** (también llamado **extensión** de la relación) es una tabla con filas y columnas. Las columnas de las relaciones corresponden a los atributos; las filas, denominadas **tuplas**, son colecciones de valores tomados de cada atributo, y desempeñan la misma función que los casos individuales de entidades en el modelo ER. El grado de una relación es el número de columnas; la **cardinalidad** de una relación es el número de tuplas. La figura 12.1 muestra un ejemplo de los conceptos citados. La relación ESTUDIANTE tiene tres atributos (NOMBRE, EDAD y SEXO) y cinco tuplas, cada una representando nombre, edad y sexo de un estudiante. Por tanto, el grado y la cardinalidad de ESTUDIANTE son tres y cinco, respectivamente.

Esquema: ESTUDIANTE (NOMBRE, EDAD, SEXO)
 Caso: ESTUDIANTE

NOMBRE	EDAD	SEXO
John Smith	19	Varón
Sally Boyce	23	Hembra
Tom Hagen	25	Varón
Bill Ballucci	20	Varón
Tina Graver	19	Hembra

Figura 12.1. Ejemplo de un esquema y caso relacional.

La definición matemática de las relaciones se desarrolla empezando por la noción de dominios. Un **dominio** es una colección de valores. Dados varios atributos, A_1, A_2, \dots, A_n , con dominios D_1, D_2, \dots, D_n , un caso de relación de grado n es simplemente un subconjunto del producto cartesiano $D_1 \times D_2 \times \dots \times D_n$. Esta definición destaca una importante propiedad de las relaciones, a saber, que son conjuntos de tuplas en el sentido matemático: una relación en ningún momento puede tener tuplas duplicadas. Sin embargo, la mayoría de los sistemas relacionales no imponen esta restricción, ya que en diversas situaciones pueden ocurrir duplicados, y puede ser útil mantenerlos. En términos estrictos, tampoco importa el orden de las tuplas en una relación.

En el modelo relacional, el concepto de clave está definido de una manera muy similar al concepto de identificador en el modelo ER; una **clave** de una relación es un conjunto de atributos de la relación que identifica de manera única cada tupla de cada extensión de esa relación. Así, la única diferencia entre nuestro uso de identificadores y claves es que en el modelo relacional sólo se acepta la identificación interna.

En general, una relación puede tener más de una clave, y cada clave se denomina **clave candidata**. Por ejemplo, la relación PERSONA puede tener dos claves candidatas: la primera un número de seguro social (NSS); la segunda una clave compuesta formada por (APELIDO, NOMBRE). Es común designar una de las claves como **clave primaria** de la relación. Nuestro convencionalismo es subrayar aquellos atributos que componen la clave primaria.

La sencillez del modelo relacional proviene del hecho de que todas las relaciones se definen de manera independiente; no hay conceptos como los de jerarquía, conexión o enlace entre las relaciones en el modelo. Sin embargo, las interrelaciones del modelo ER se pueden representar en el modelo relacional por una operación explícita de *equirreunión (equi-join)* entre atributos de tablas diferentes. Si realizamos una *reunión (join)*, que es la operación del álgebra relacional para hacer correspondencias entre dos tablas, la correspondencia debe hacerse entre atributos comparables, esto es, atributos en el mismo dominio. El ejemplo de la figura 12.2 muestra tres relaciones independientes, CURSO, ES-

NOMBRE	EDAD	SEXO	CODIGO	INSTRUCTOR
John Smith	19	Varón	CS347	Ceri
Sally Boyce	23	Hembra	CS311	Batini
Tom Hagen	25	Varón	CS144	Navathe
Bill Ballucci	20	Varón		
Tina Graver	19	Hembra		

NUMERO_CURSO	NOMBRE_ESTUDIANTE	CALIFICACION
CS347	John Smith	A+
CS347	Sally Boyce	B-
CS311	Tom Hagen	A
CS144	John Smith	B+
CS144	Sally Boyce	A-

Figura 12.2. La relación EXAMEN como correspondencia de valores explícita entre las relaciones ESTUDIANTE y CURSO.

ESTUDIANTE y EXAMEN, para la base de datos escolar. La interrelación de uno a muchos entre ESTUDIANTE y EXAMEN se obtiene igualando los atributos NOMBRE de ESTUDIANTE y NOMBRE_ESTUDIANTE de EXAMEN. Obsérvese que por cada caso de ESTUDIANTE existen cero, uno o más casos de EXAMEN, relacionados por el mismo nombre de estudiante. De manera similar, la interrelación de uno a muchos entre CURSO y EXAMEN se obtiene igualando los atributos CODIGO de CURSO y NUMERO_CURSO de examen.

12.1.1. Restricciones de integridad en los esquemas relacionales de bases de datos

Ahora veremos las restricciones de integridad que pueden ser especificadas en un esquema relacional. Se espera que tales restricciones, una vez especificadas, se cumplan para cada caso de base de datos de ese esquema. Sin embargo, en los productos comerciales de DBMS actuales no siempre pueden ser especificadas todas esas restricciones. Además, aun especificadas, no se obliga automáticamente el cumplimiento de todas ellas. Se consideran tres tipos de restricciones como parte del modelo relacional: restricciones de clave, de integridad de entidades y de integridad referencial. Las **restricciones de clave** especifican las claves candidatas de cada esquema de relación; los valores de las claves candidatas deben ser únicos para cada tupla en cualquier caso de ese esquema de relación. En el ejemplo de la figura 12.2, NOMBRE es una clave para ESTUDIANTE, CODIGO es una clave para CURSO, y el par (NUMERO_CURSO, NOMBRE_ESTU-

DIANTE) es una clave para examen. Estas constituyen las restricciones de clave en el esquema de base de datos compuesto por las tres relaciones.

Las **restricciones de integridad de entidades** establecen que ningún valor de clave primaria puede ser nulo. Esto es porque el valor de la clave primaria se usa para identificar las tuplas individuales de una relación; permitir valores nulos para la clave primaria implica que *no se pueda identificar* algunas tuplas. Por ejemplo, si dos o más tuplas tuvieran un valor nulo como clave primaria, no podríamos distinguirlas. En el ejemplo de la figura 12.2, debido a las restricciones de integridad de entidades, NOMBRE no puede ser nulo en ninguna tupla de ESTUDIANTE, CODIGO no puede ser nulo en ninguna tupla de CURSO, y el par (NUMERO_CURSO, NOMBRE_ESTUDIANTE) debe tener ambos valores no nulos en cualquier tupla de EXAMEN.

Las restricciones de clave y de integridad de la entidad se especifican en relaciones individuales. La **restricción de integridad referencial**, en cambio, se especifica entre dos relaciones, y se usa para mantener la congruencia entre las tuplas de las dos relaciones. Informalmente, la restricción de integridad referencial establece que una tupla de una relación que haga referencia a otra relación debe referirse a una tupla existente en esa relación.

Para definir la integridad referencial de manera más formal, primero se debe definir el concepto de clave ajena. Las condiciones para una clave ajena, dadas a continuación, especifican una restricción de integridad referencial entre los dos esquemas de relación R_1 y R_2 . Un conjunto de atributos CLA del esquema de relación R_1 es una clave ajena de R_1 si satisface estas dos reglas:

1. Los atributos de CLA tienen el mismo dominio que los atributos de la clave primaria, CLP, de otro esquema de relación R_2 ; se dice que los atributos de CLA **se refieren a** la relación R_2 .
2. Un valor de CLA para una tupla t_1 de R_1 existe como valor de CLP para alguna tupla t_2 de R_2 , o bien es nulo. En el primer caso, tenemos $t_1[CLA] = t_2[CLP]$, y decimos que la tupla t_1 se refiere a la tupla t_2 . (Nota: $t[X]$ se refiere al valor del atributo X de la tupla t. X puede ser un grupo de atributos.)

En una base de datos de muchas relaciones, habrá normalmente muchas restricciones de integridad referencial que correspondan a las interrelaciones de las entidades representadas por las relaciones. Por ejemplo, en la base de datos que se muestra en la figura 12.2, la relación EXAMEN tiene la clave primaria (NUMERO_CURSO, NOMBRE_ESTUDIANTE). El atributo NOMBRE_ESTUDIANTE se refiere al estudiante que hizo el examen, por tanto, se puede designar NOMBRE_ESTUDIANTE como clave ajena de EXAMEN, refiriéndose a la relación ESTUDIANTE. Esto significa que un valor de NOMBRE_ESTUDIANTE en cualquier tupla t_1 de la relación EXAMEN debe: 1) coincidir con un valor de la clave primaria de ESTUDIANTE (el atributo NOMBRE) en alguna tupla t_2 de la relación ESTUDIANTE, o bien 2) ser nulo. Sin embargo, un valor nulo de NOMBRE_ESTUDIANTE no está permitido porque viola la restricción de integridad de entidades de la relación EXAMEN. Supóngase que se tuviera una clave ajena denominada NOMBRE_EDIFICIO en EXAMEN (especificando el lugar del examen), que se refi-

riera a la clave primaria de una relación denominada EDIFICIO; de nuevo habría una restricción referencial, pero ahora se permitirían valores nulos de NOMBRE_EDIFICIO en EXAMEN (especificando que el lugar del examen se desconoce).

12.2. Correspondencia de esquemas del modelo ER al modelo relacional

Este apartado presenta una metodología para el diseño lógico que tiene como objetivo el modelo relacional. Se supone que el punto de partida es un esquema lógico ER similar al producido en la Figura 11.15; el resultado de la correspondencia es un esquema relacional. Este consiste en un conjunto de definiciones de relaciones, en el cual cada relación tiene una clave primaria. Las relaciones producidas por la transformación de esquemas corresponden a entidades o bien a interrelaciones, y mantienen la misma forma normal. El concepto de **forma normal** fue propuesto en la literatura relacional como una medida de la ausencia de las anomalías o dificultades que surgen cuando se trata de insertar, eliminar o modificar registros de las bases de datos relacionales. Cuanto más alta sea la forma normal, mayor será la «pureza» o «bondad» de la definición lógica del esquema relacional. Si el lector desea una explicación detallada de las formas normales, puede consultar el capítulo 6; si desea un tratamiento de las formas normales en el contexto del modelo relacional, puede leer el capítulo 13 del libro de texto de Elmasri y Navathe (véase la bibliografía del capítulo 1).

La metodología propuesta en esta sección convierte un esquema ER en un conjunto de entidades e interrelaciones, tales que su correspondencia con el modelo relacional sea sencilla. Esta «correspondencia preparatoria» consiste en dos actividades: 1) la eliminación de identificadores externos (este paso también se asocia con la eliminación de algunas interrelaciones), y 2) la eliminación de atributos compuestos y polivalentes del esquema.

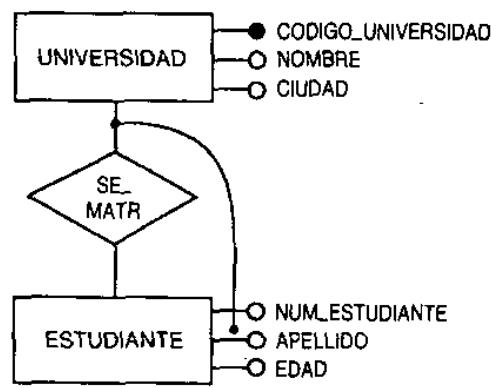
Una vez completada esta «correspondencia preparatoria», se está en condiciones de aplicar los siguientes pasos:

1. Transformación de cada entidad del esquema en una relación.
2. Transformación de cada interrelación: las interrelaciones de muchos a muchos requieren una relación individual, mientras que las interrelaciones de uno a uno o de uno a muchos pueden ser modeladas añadiendo atributos a las relaciones existentes.

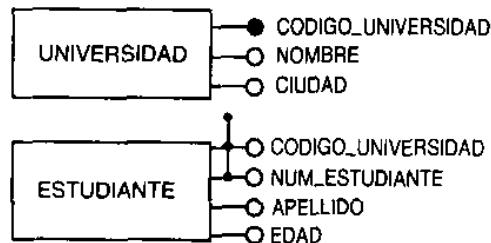
A continuación se considerarán estas actividades en orden.

12.2.1. Eliminación de identificadores externos

Como no se puede usar identificadores externos en el modelo relacional, se deben transformar en identificadores internos. Supongamos que la clave primaria



(a) Esquema inicial



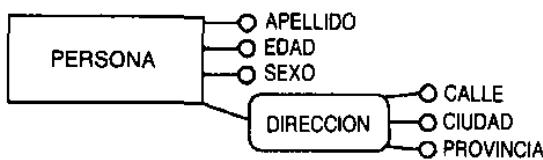
(b) Esquema final

Figura 12.3. Eliminación de un identificador externo.

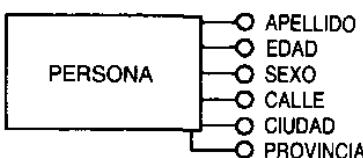
de una entidad E_1 , tal como se seleccionó en el apartado 11.7, es un identificador externo o mixto, y sea la entidad E_2 la que suministra la identificación externa a través de la interrelación R a E_1 ; supondremos además que E_2 tiene un identificador interno como su clave primaria. Para eliminar el identificador externo de E_1 , se debe importar a la entidad E_1 la clave primaria de E_2 . Después de esta operación, puede eliminarse la interrelación R ; de hecho, el vínculo entre E_1 y E_2 se modela al insertar en E_1 la clave primaria de E_2 .

Considérese el ejemplo de la figura 12.3a; se representa la identificación externa, mixta, de la entidad ESTUDIANTE (que tiene un identificador externo suministrado por la entidad UNIVERSIDAD) al incluir CÓDIGO_UNIVERSIDAD en la entidad ESTUDIANTE. De esta manera, la interrelación SE_MATRICULA queda modelada automáticamente en virtud de la clave primaria (CÓDIGO_UNIVERSIDAD, NUM_ESTUDIANTE) de la entidad ESTUDIANTE, y se puede eliminar del esquema. El resultado se muestra en la figura 12.3b.

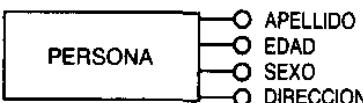
Consideremos otra vez la identificación externa de una entidad genérica E_1 desde la entidad E_2 ; el procedimiento descrito anteriormente no sería posible si E_2 a su vez estuviera identificada externamente por alguna otra entidad E_3 . Este proceso debe realizarse empezando por las entidades que tienen como clave primaria un identificador interno (denominadas entidades fuertes) y luego conti-



(a) Esquema con un atributo compuesto



(b) Atributo compuesto reducido a sus componentes



(c) Atributo compuesto considerado como atributo simple

Figura 12.4. Eliminación de un atributo compuesto.

nuando con las entidades vecinas. Los identificadores pueden propagarse según se necesite para la identificación externa. Así, en el presente ejemplo, supongamos que E_3 tiene un identificador interno como clave primaria. Los identificadores «fluirán» entonces de E_3 a E_2 y de E_2 a E_1 . La selección de una sola clave primaria en el apartado 11.7 es muy útil porque obliga a concretar el identificador que deberá ser propagado a otras entidades cuando se haga la correspondencia en las relaciones.

12.2.2. Eliminación de atributos compuestos y polivalentes

El modelo relacional en su forma básica contiene sólo atributos simples, monovalentes; por ello, los atributos compuestos y polivalentes deben ser modelados en términos de atributos simples monovalentes. Con cada atributo compuesto, se tienen básicamente dos alternativas: 1) eliminar el atributo compuesto considerando todos sus componentes como atributos individuales, o 2) eliminar los componentes individuales y considerar el atributo compuesto entero como un solo atributo.

Las dos alternativas se ilustran en las figuras 12.4b y 12.4c, respectivamente, para el atributo compuesto DIRECCION de la entidad PERSONA. En el primer caso, se pierde el concepto de que CALLE, CIUDAD y ESTADO son atributos relaciona-

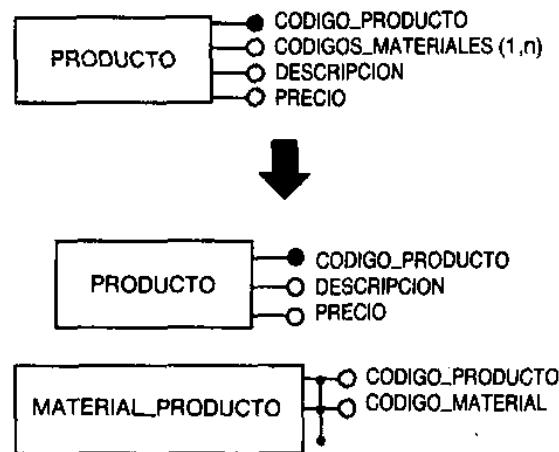


Figura 12.5. Eliminación de un atributo polivalente de una entidad.

dos. En el último caso se pierde el concepto de que DIRECCION puede ser descompuesto en sus partes; ahora DIRECCION tiene un solo tipo, digamos una cadena de caracteres de longitud 50. Descomponerla en CALLE, CIUDAD y ESTADO se convierte en una responsabilidad de la aplicación.

Los atributos polivalentes requieren la introducción de entidades nuevas; cada atributo polivalente distinto requiere una entidad en la cual pueda estar representado como un atributo monovalente. La nueva entidad contiene el atributo polivalente más el identificador de la entidad original; el identificador de la nueva entidad es el conjunto de todos sus atributos.

En la figura 12.5 se considera la entidad PRODUCTO, con el atributo polivalente CODIGOS_MATERIALS, el cual proporciona los códigos de los diversos materiales que constituyen un producto. Para dar cuenta de este atributo polivalente, se necesita crear una entidad aparte, MATERIAL_PRODUCTO, que incluya CODIGO_MATERIAL y la clave primaria CODIGO_PRODUCTO de PRODUCTO. La clave primaria de MATERIAL_PRODUCTO es el par (CODIGO_PRODUCTO, CODIGO_MATERIAL).

Atributos polivalentes de las interrelaciones. Si el atributo polivalente pertenece a una interrelación R entre las entidades E₁ y E₂, se necesita crear una entidad nueva NE para representarlo. La nueva entidad NE incluye uno o dos atributos tomados de E₁, E₂, o ambos, dependiendo del tipo de la interrelación:

1. Si la interrelación es de uno a uno, NE incluye la clave primaria de E₁ o bien de E₂.
2. Si la interrelación entre E₁ y E₂ es de uno a muchos, NE incluye la clave primaria de E₂ (suponiendo que E₂ está en el lado de «muchos»).
3. Si la interrelación entre E₁ y E₂ es de muchos a muchos, NE incluye las claves primarias tanto de E₁ como de E₂.

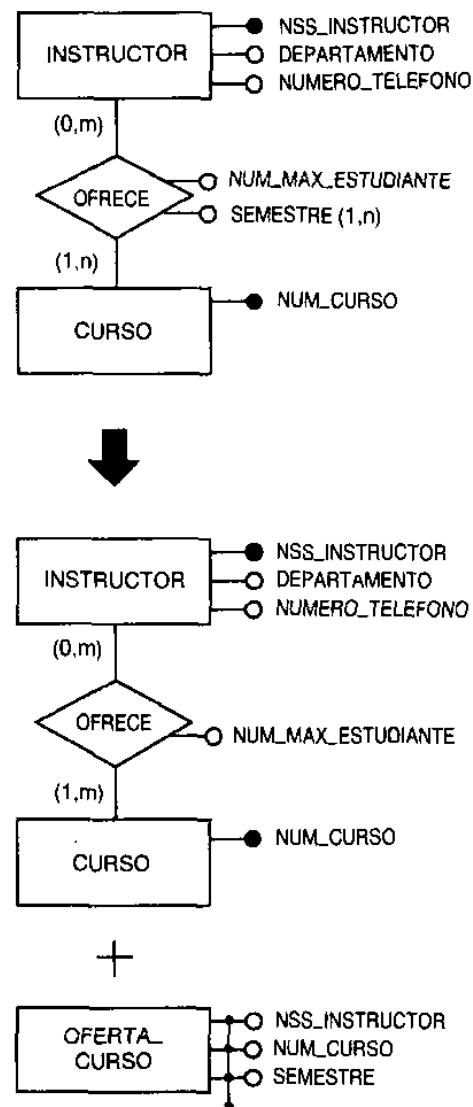


Figura 12.6. Eliminación de un atributo polivalente de una interrelación creando una entidad aparte.

La clave primaria de NE está constituida por todos sus atributos: esto incluye aquellos que se «tomaron prestados» de E₁ y E₂, y el atributo polivalente. Cualquier atributo no polivalente que tenga R seguirá siendo atributo de R. Se muestra en la figura 12.6 un ejemplo del caso 3 para atributos polivalentes de las interrelaciones. Cada instructor ofrece muchos cursos. La interrelación OFRECE tiene dos atributos: NUM_MAX_ESTUDIANTES, que es monovalente, y SEMESTRE, que es polivalente. Para resolver semestre, se crea una nueva entidad, OFERTA_CURSO, con los siguientes atributos: NSS_INSTRUCTOR de INSTRUCTOR, NUM_CURSO de CURSO, y el atributo *monovalente* SEMESTRE. La clave primaria de la nueva entidad OFERTA_CURSO contiene todos sus atrí-

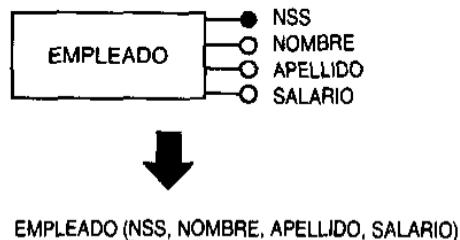


Figura 12.7. Transformación de una entidad.

butos. Ahora se puede continuar con la correspondencia como se indica en seguida.

12.2.3. Transformación de entidades

Este paso es bastante simple: se transforma cada entidad del esquema en una relación. Los atributos y la clave primaria de la entidad se convierten en los atributos y la clave primaria de la relación. Un ejemplo se muestra en la figura 12.7.

12.2.4. Transformación de interrelaciones de uno a uno

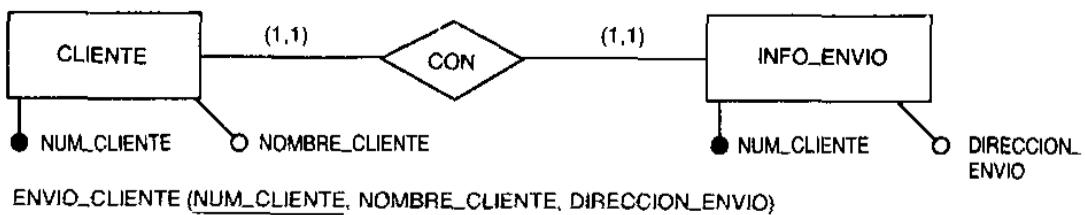
Ahora debemos tratar de las interrelaciones. Se empieza por considerar las interrelaciones binarias de una manera general. Se verán las interrelaciones de uno a uno, de uno a muchos y de muchos a muchos de manera individual. El proceso de transformación es también influido por las cardinalidades mínimas de las dos entidades que participan en la interrelación.

En principio, las dos entidades E_1 y E_2 que participan en la interrelación producen relaciones individuales; de otro modo se habrían fusionado durante el diseño lógico independiente del modelo. Respecto a la interrelación, hay que distinguir si las dos entidades E_1 y E_2 tienen una participación total en la interrelación, o si una o ambas tienen una participación parcial en la misma. Así, tenemos los siguientes casos:

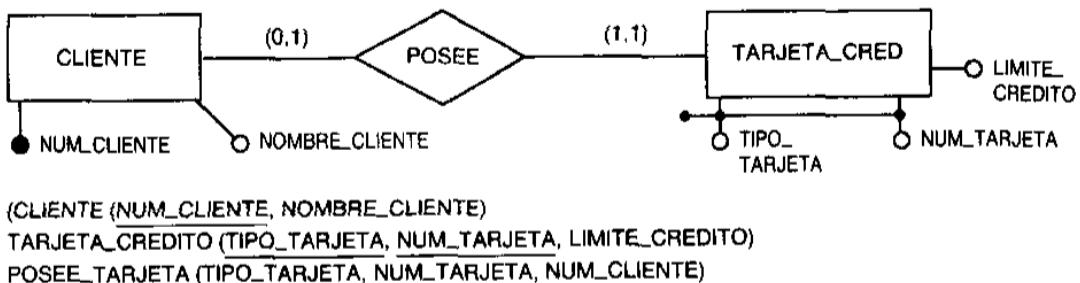
Integración en una relación. Esta opción tiene sentido cuando la participación de las dos entidades en la interrelación es total. Hay dos posibilidades:

Caso 1: Las dos entidades tienen las mismas claves primarias. Supóngase que tanto CLIENTES como INFO_ENVIO tienen la clave primaria NUM_CLIENTE. En este caso, las dos relaciones correspondientes se integran en una relación combinando todos los atributos e incluyendo la clave primaria sólo una vez. Este caso se ilustra en la figura 12.8a.

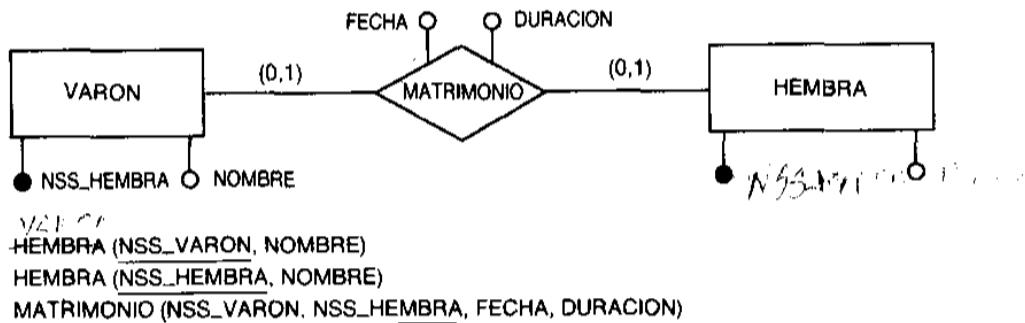
Caso 2: Las dos entidades tienen claves primarias diferentes: Supóngase que



(a) Integración en una relación



(b) Relaciones individuales; una entidad tiene participación parcial



(c) Relaciones individuales, ambas tienen participación parcial

Figura 12.8. Transformación de una relación de uno a uno: diferentes restricciones de participación.

CLIENTE e **INFO_ENVIO** tienen diferentes claves primarias, digamos **NUM_CLIENTE**, y (**CODIGO_POSTAL**, **CALLE**, **NUM_CASA**), respectivamente. En este caso también se integran en una relación combinando todos los atributos e incluyendo las claves primarias de ambas. Una de las dos claves primarias será la clave primaria de la relación resultante; por ejemplo, en la relación que sigue se escoge **NUM_CLIENTE**.

ENVIO_CLIENTE (**NUM_CLIENTE**, **NOMBRE_CLIENTE**, **NUM_CASA**, **CALLE**, **CODIGO_POSTAL**)

Definición de una relación aparte. Esta opción se usa cuando una o las dos entidades tienen una participación parcial. Un ejemplo de cada caso se muestra en las figuras 12.8b y 12.8c.

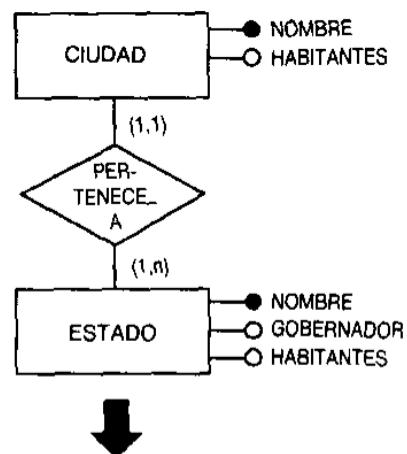
Caso 1: Una entidad con participación parcial. Esto se refiere, por ejemplo, a los clientes de un banco, a los cuales el banco emite *cero* o *una* tarjetas de crédito. En la figura 12.8b, cada tarjeta de crédito *debe* pertenecer a un cliente, pero un cliente puede no tener tarjeta de crédito. En este caso, las dos relaciones, CLIENTE, y TARJETA_DE_CREDITO, ya han sido creadas. Se define una relación adicional POSEE_TARJETA (NUM_CLIENTE, TIPO_TARJETA, NUM_TARJETA) usando la clave primaria de las dos relaciones. Tanto NUM_CLIENTE como (TIPO_TARJETA, NUM_TARJETA) son claves candidatas de POSEE_TARJETA, y por consiguiente pueden ser declaradas como la clave primaria. Obsérvese que se puede usar la primera opción en este caso y representar todo en una sola relación. En ese caso se debe escoger NUM_CLIENTE como clave primaria de la relación integrada; aquellos clientes que no posean tarjeta de crédito tendrán valores nulos de los atributos TIPO_TARJETA, NUM_TARJETA. No se puede seleccionar (TIPO_TARJETA, y NUM_TARJETA) como clave primaria de la relación integrada, porque en este caso los clientes sin tarjeta de crédito no podrían ser representados.

Caso 2: Las dos entidades con participación parcial. Considérese la interrelación MATRIMONIO entre las entidades VARON y HEMBRA. En este caso ambas tienen una participación parcial en la interrelación MATRIMONIO. Para evitar valores nulos y representar tanto las entidades como la interrelación, se crea la relación MATRIMONIO (NSS_VARON, NSS_HEMBRA, FECHA, DURACION) además de las relaciones VARON y HEMBRA.

12.2.5. Transformación de interrelaciones de uno a muchos

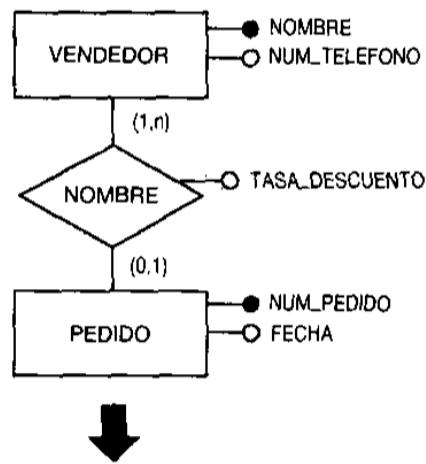
Considérese la interrelación R entre dos entidades, E_1 y E_2 ; supongamos que R es una interrelación de uno a muchos. En este caso, se dará cuenta de la interrelación incluyendo la clave primaria de E_1 en la relación correspondiente a E_2 como uno o más atributos simples. Obsérvese que ya se ha dado cuenta de los identificadores externos. Por consiguiente, esta transferencia de la clave no tiene propósitos de identificación. Los posibles atributos de la interrelación tienen que ser trasladados a la relación que modela la entidad E_2 . Otra vez son posibles dos casos:

Caso 1: La entidad en el lado de «muchos» tiene una participación obligatoria. Esto está ejemplificado en la figura 12.9a, donde hay una interrelación de uno a muchos entre ESTADO y CIUDAD, y CIUDAD tiene una participación total



CIUDAD (NOMBRE_CIUDAD, NOMBRE_ESTADO, HABITANTES)
 ESTADO (NOMBRE_ESTADO, GOBERNADOR, HABITANTES)

(a) Participación total



VENDEDOR (NOMBRE, NUM_TELEFONO)
 PEDIDO (NUM_PEDIDO, FECHA)
 PEDIDO_VENTAS (NUM_PEDIDO, NOMBRE_VENDEDOR, TASA_DESCUENTO)

(b) Participación parcial

Figura 12.9. Transformación de una interrelación de muchos a muchos.

en la interrelación; por tanto, la clave primaria NOMBRE_ESTADO de ESTADO se incluye en la relación CIUDAD.

Caso 2: La entidad en el lado de «muchos» tiene una participación parcial. En la figura 12.9b hay una interrelación entre VENDEDOR y PEDIDO. Supóngase

que los pedidos pueden hacerse por medio de los vendedores, en cuyo caso se aplica una tasa de descuento, y también directamente sin vendedores, sin aplicar una tasa de descuento. Así pues, existe la posibilidad de valores nulos de NOMBRE_VENDEDOR y TASA_DESCUENTO en la relación PEDIDO si se usan las siguientes correspondencias:

VENDEDOR (NOMBRE, NUM_TELEFONO)
PEDIDO (NUM_PEDIDO, FECHA, NOMBRE_VENDEDOR, TASA_DESCUENTO)

Si el número relativo de esos pedidos es grande, y no se puede admitir valores nulos, una mejor alternativa sería establecer tres relaciones (lo cual es el caso más general):

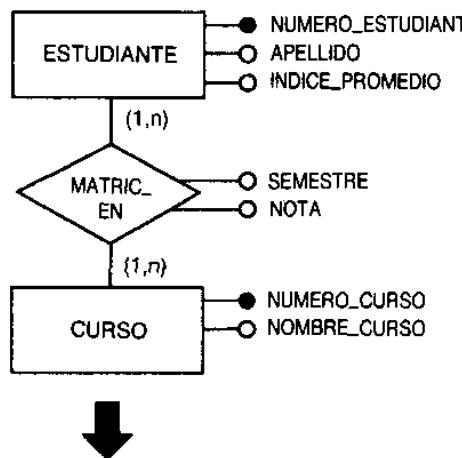
VENDEDOR (NOMBRE, NUM_TELEFONO)
PEDIDO (NUM_PEDIDO, FECHA)
PEDIDO_VENTAS (NUM_PEDIDO, NOMBRE_VENDEDOR, TASA_DESCUENTO)

Obsérvese que las dos relaciones, PEDIDO y PEDIDO_VENTAS, describen pedidos. La primera atañe a *todos* los pedidos; PEDIDO_VENTAS contiene un subconjunto de todos los pedidos, aquéllos hechos a través de vendedores. Así, se tiene la restricción adicional de que el conjunto de números de pedidos en PEDIDO_VENTAS esté siempre incluido en el conjunto de números de pedidos de la relación PEDIDO. Esta interrelación se denomina **dependencia de inclusión** de NUM_PEDIDO en PEDIDO_VENTAS respecto a NUM_PEDIDO en PEDIDO en el modelo relacional¹.

12.2.6. Transformación de interrelaciones de muchos a muchos

En el caso de interrelaciones de muchos a muchos, la solución no depende de la cardinalidad mínima de la interrelación. Supongamos que R es una interrelación de muchos a muchos entre E₁ y E₂. Se crea una relación nueva que tiene como clave primaria la combinación de atributos que constituyen las claves primarias tanto de E₁ como de E₂, y que incluye como atributos los atributos de R. En el ejemplo de la figura 12.10, una interrelación de muchos a muchos MATRICULADO_EN entre ESTUDIANTE y CURSO se modela como una nueva relación MATRICULADO_EN, que tiene como clave primaria el par (NUMERO_ESTUDIANTE, NUMERO_CURSO), con SEMESTRE y NOTA como atributos. Obsérvese que NUMERO_ESTUDIANTE y NUMERO_CURSO son claves ajenas y tienen restricciones referenciales respecto a las claves primarias correspondientes.

¹ La dependencia de inclusión R.X < S.Y se lee como «el atributo X de la relación R presenta dependencia de inclusión respecto al atributo Y de la relación S». Implica que, en cualquier momento, la proyección de R en X es un subconjunto de la proyección de S en Y.



ESTUDIANTE (NUMERO_ESTUDIANTE, APELLIDO, INDICE_PROMEDIO)
CURSO (NUMERO_CURSO, NOMBRE_CURSO)
MATRICULADO_EN (NUMERO_ESTUDIANTE, NUMERO_CURSO, SEMESTRE, NOTA)

Figura 12.10. Transformación de una interrelación de muchos a muchos.

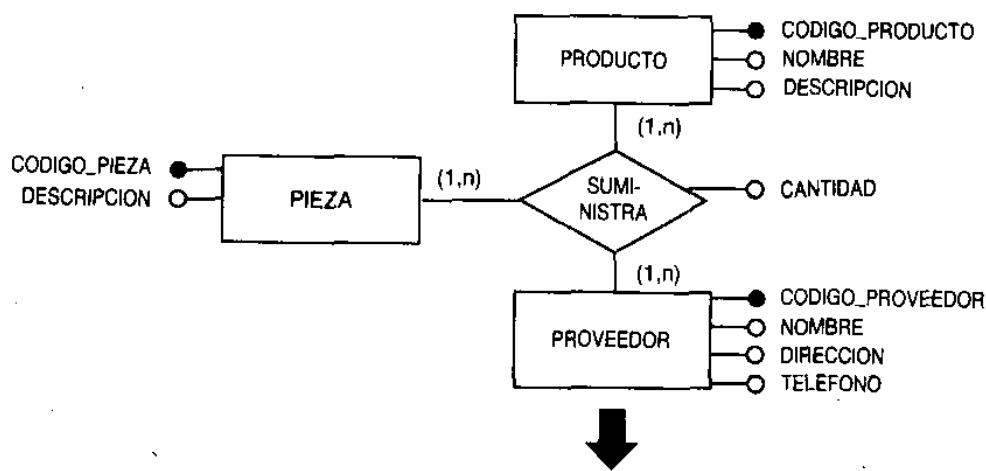
12.2.7. Transformación de interrelaciones n-arias y recursivas

Finalmente, hay que resolver los dos tipos de interrelaciones que quedan: las n-arias ($n > 2$) y las recursivas. Las interrelaciones n-arias siguen las mismas reglas de transformación que las binarias de muchos a muchos: la relación hereda todos los identificadores de las n entidades (que forman la clave de la nueva relación). En algunos casos especiales, la clave obtenida de esta manera no es mínima, y un subconjunto de claves primarias es de hecho la clave mínima. La clave mínima es aquella que no contiene ninguna dependencia funcional (véase el apartado 6.5.1) entre sus atributos. La figura 12.11a muestra la interrelación ternaria SUMINISTRA entre PRODUCTO, PIEZA, y PROVEEDOR; la clave de la relación SUMINISTRA es el trío (CÓDIGO_PRODUCTO, CÓDIGO_PIEZA, CÓDIGO_PROVEEDOR). Esta clave no puede reducirse más.

Como ejemplo de la reducción de la clave por defecto en una correspondencia así, considérese la interrelación cuaternaria VENTA_COCHE entre COCHE, CLIENTE, COMERCIANTE y BANCO_FINANCIADOR de la figura 12.11b. La relación resultante tiene la siguiente clave primaria por defecto:

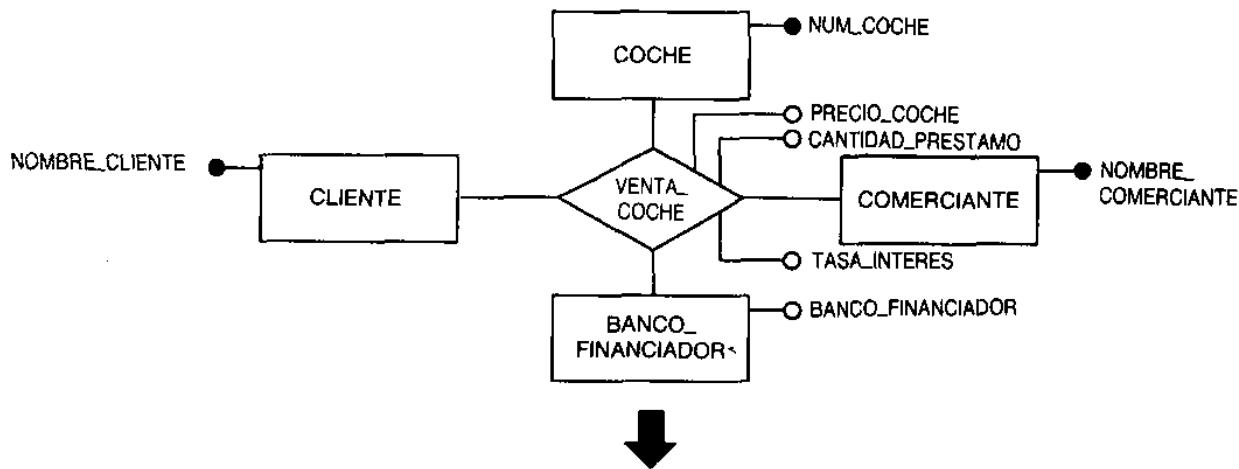
VENTA_COCHE (NUM_COCHE, NOMBRE_CLIENTE, NOMBRE_COMERCIANTE, NOMBRE_BANCO, PRECIO_COCHE, IMPORTE_PRESTAMO, TASA_INTERES)

Sin embargo, si un coche pertenece estrictamente a un cliente, NUM_COCHE determina funcionalmente a NOMBRE_CLIENTE, y éste puede ser retirado de la clave. De manera similar, si un comerciante usa solamente un banco para el



PRODUCTO (CODIGO_PRODUCTO, NOMBRE, DESCRIPCION)
PIEZA (CODIGO_PIEZA, DESCRIPCION)
PROVEEDOR (CODIGO_PROVEEDOR, NOMBRE, DIRECCION, TELEFONO)
SUMINISTRA (CODIGO_PRODUCTO, CODIGO_PIEZA, CODIGO_PROVEEDOR, CANTIDAD)

(a) Una interrelación ternaria



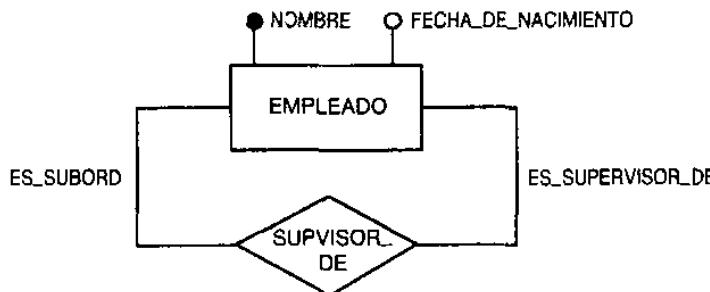
VENTA_COCHE (NUM_COCHE, NOMBRE_CLIENTE, NOMBRE_COMERCIANTE, NOMBRE_BANCO,
PRECIO_COCHE, CANTIDAD_PRESTAMO, TASA_INTERES)

(b) Una interrelación cuaternaria

Figura 12.11. Transformación de interrelaciones n-arias.

financiamiento, **NOMBRE_COMERCIANTE** determina funcionalmente a **NOMBRE_BANCO**, y éste puede ser retirado de la clave.

Una interrelación recursiva R de una entidad E a sí misma se modela como una nueva relación que incluye dos atributos; ambos corresponden a la clave



Interrelación de muchos a muchos:

EMPLEADO (NOMBRE, FECHA_DE_NACIMIENTO)

SUPERVISOR_DE (NOMBRE_DE_SUPERVISOR, NOMBRE_DE_SUBORDINADO)

Interrelación de uno a muchos:

EMPLEADO (NOMBRE, FECHA_DE_NACIMIENTO)

SUPERVISOR_DE (NOMBRE_DE_SUBORDINADO), NOMBRE_DE_SUPERVISOR

o bien

EMPLEADO (NOMBRE, FECHA_DE_NACIMIENTO, NOMBRE_DE_SUPERVISOR)

Figura 12.12. Transformación de una interrelación recursiva.

primaria de E, y sus nombres corresponden a los dos papeles de E en la interrelación. Uno de ellos (o ambos) se elige(n) como clave primaria para la nueva relación, de acuerdo con el tipo de la interrelación (uno a uno, uno a muchos, muchos a muchos), como se expuso anteriormente. En la figura 12.12 la interrelación cíclica SUPERVISOR_DE se transforma en una relación SUPERVISOR_DE, cuyos atributos son NOMBRE_DE_SUPERVISOR y NOMBRE_DE_SUBORDINADO (que son versiones rebautizadas de los atributos NOMBRE de EMPLEADO). Si un empleado puede tener varios supervisores, la interrelación es de muchos a muchos, y la clave primaria de SUPERVISOR_DE contiene los dos atributos. Por otra parte, si un empleado sólo puede tener un supervisor, la interrelación es de uno a muchos; la clave primaria de SUPERVISOR_DE es entonces sólo NOMBRE_DE_SUBORDINADO. Como esta clave es la misma que NOMBRE en EMPLEADO, la relación SUPERVISOR_DE es «similar» a EMPLEADO y puede ser integrada en EMPLEADO, con NOMBRE_DE_SUPERVISOR como atributo adicional. Esta última solución de una sola relación será en general la preferida, a menos que un empleado pueda existir sin supervisor, porque entonces se tendrían valores nulos de NOMBRE_DE_SUPERVISOR para algunos empleados.

En este punto se ha completado la correspondencia del esquema lógico ER al modelo relacional. Todos los rasgos del esquema ER están modelados como relaciones con una clave primaria.

12.3. Correspondencia de operaciones del modelo ER al modelo relacional

En este apartado se muestran varios ejemplos de transformación de especificaciones de esquemas de navegación al lenguaje relacional más popular: SQL. Se supone que el lector está familiarizado con SQL.

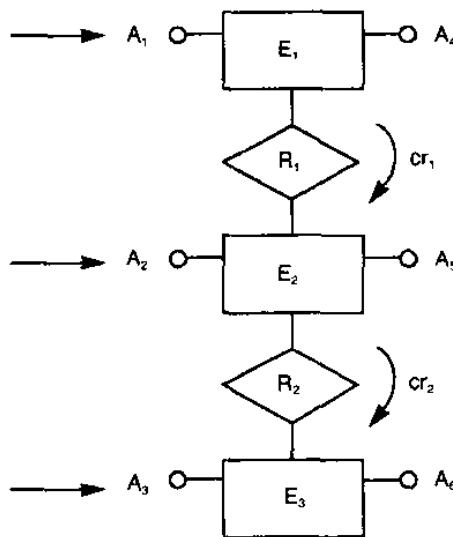
La correspondencia de un esquema de navegación con SQL sigue esta heurística general:

1. Si se recupera una entidad completa, se modela con el enunciado SELECT*; de lo contrario, la lista de atributos se muestra como lista de SELECT.
2. La navegación de una entidad a otra a través de una interrelación resulta en una operación de reunión; los nombres de las relaciones apropiadas se incluyen en la lista FROM.
3. Las condiciones relativas a una entidad (denominadas *condiciones de selección*), o las condiciones para «cotejar» entidades (denominadas *condiciones de reunión*) aparecen en la cláusula WHERE. Esta cláusula puede hacerse compleja y contener bloques SELECT FROM WHERE inmersos cuando la consulta requiere condiciones de calificación adicionales.
4. Las cláusulas GROUP BY y HAVING son necesarias cuando se desea realizar agregaciones para las cuales no se tiene una notación fija en el esquema de navegación.

La figura 12.13 muestra un esquema de una consulta de navegación en el modelo ER y su correspondencia en SQL. La consulta que se muestra se conoce en general como consulta de *selección-proyección-reunión* porque la ejecución de la consulta en SQL requiere estas tres operaciones estándar del álgebra relacional. Daremos dos ejemplos. En la figura 12.14 se muestra la especificación de navegación de una operación simple sobre la entidad INSTRUCTOR. Obsérvese que la condición «menos de 30 años de edad» corresponde a una selección por EDAD, que aparece en la cláusula WHERE. En la figura 12.15 el esquema de navegación requiere obtener acceso a CURSO mediante una navegación de INSTRUCTOR a CURSO a través de OFRECE. Esto se traduce en una operación de reunión (*join*) entre las relaciones CURSO y OFRECE. Se muestran dos formas de las consultas en SQL: una con un solo bloque que usa una cláusula WHERE, la cual tiene una combinación de las condiciones de selección y de reunión ; la otra con un anidamiento de dos bloques de SQL. El parámetro de entrada se muestra como la variable X precedida por un signo de \$.

12.4. Base de datos del caso de estudio en el modelo relacional

En este apartado se continuará con el caso de estudio de los capítulos anteriores. Primero se aplicará la metodología mostrada en el apartado 12.2 para ge-

(a) Esquema de navegación con COND (A_1, A_2, A_3) como condición de entrada

```

SELECT      A4, A5, A6
FROM        E1, E2, E3
WHERE       cr1 AND cr2 AND COND (A1, A2, A3)
  
```

(b) Su representación equivalente en SQL

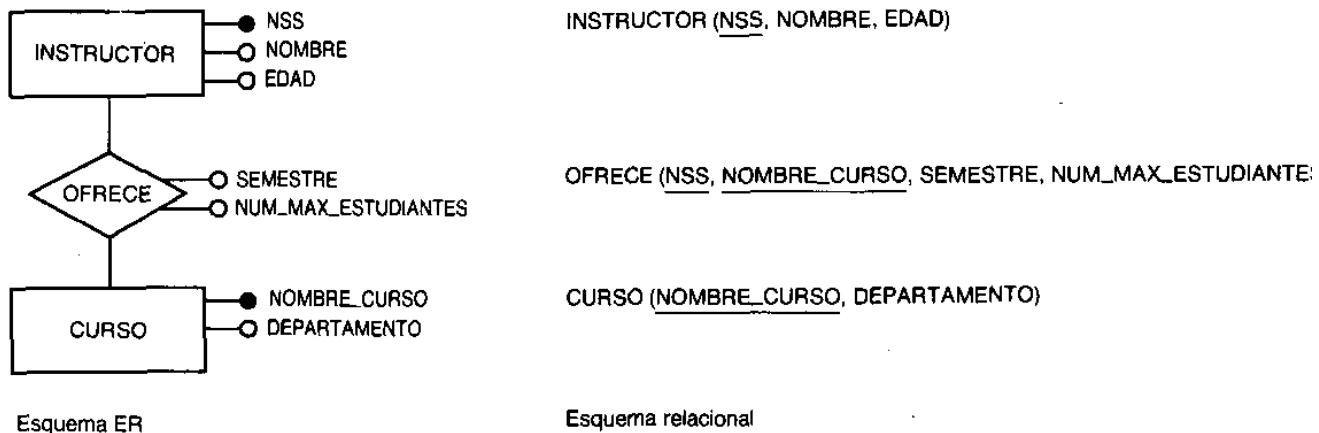
Figura 12.13. Esquema de una consulta de navegación en el modelo ER y su correspondencia en SQL.

nerar el esquema lógico relacional, y luego se mostrará la transformación de algunas operaciones de navegación a SQL.

12.4.1. Correspondencia de esquemas con el modelo relacional

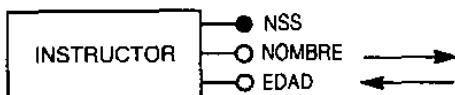
La correspondencia del esquema conceptual a lógico (datos para el caso de estudio) de la figura 11.15 con un esquema relacional es bastante sencilla. Obsérvese que para ser congruentes con el análisis del capítulo 11 se excluye la parte del esquema encerrada dentro de la línea punteada en esa figura. Esto se hace para limitar el tamaño de ejemplo; además, la parte que está dentro de esa línea es menos relevante para la aplicación principal.

Se comenzará por modelar la identificación externa de las entidades. Recuérdense las selecciones que se hicieron para eliminar los identificadores externos en el análisis del caso de estudio del apartado 11.8. En concreto, se eliminó la identificación externa de SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP suministrada por la entidad VIAJE al incluir el atributo NU-



Consulta 1: Encontrar instructores que tienen menos de 30 años de edad

Especificación de navegación en ER



Especificación SQL

```
SELECT NOMBRE
FROM INSTRUCTOR
WHERE EDAD < 30
```

Figura 12.14. Especificación de navegación y SQL de una recuperación condicional de un tipo de entidad.

MERO_VIAJE tanto en SEGMENTO_DE_RECORRIDO_ORD como en SEGMENTO_DE_RECORRIDO_POP. En consecuencia, el identificador de estas entidades es el par (NUMERO_VIAJE, NUMERO_SEGMENTO).

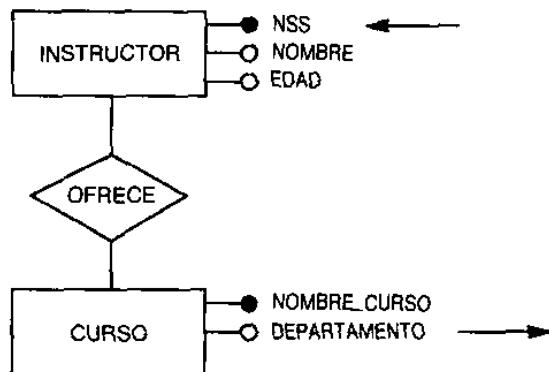
Se había realizado una eliminación similar de la identificación externa de SEGMENTO_DE_RECORRIDO_DIARIO al incluir en él los atributos NUMERO_VIAJE y NUMERO_SEGMENTO; en consecuencia, la clave primaria de SEGMENTO_DE_RECORRIDO_DIARIO era el trío (NUMERO_VIAJE, NUMERO_SEGMENTO, FECHA).

Obsérvese que, después de estas transformaciones, ya se había dado cuenta de las relaciones DE_S_R_POP, DE_S_R_ORD, POP_EN y ORD_EN y, por consiguiente, ahora pueden ser eliminadas. No hay atributos polivalentes o compuestos. Cada entidad corresponde a una relación.

Finalmente, hay que transformar las interrelaciones restantes. Las interrelaciones de uno a uno ES_UN_PASAJERO y ES_UNA_AGENCIA se modelan integrándolas en una relación, lo que también da cuenta de las entidades participantes (véase la primera opción en el apartado 12.2.4). Esto hace que surjan dos relaciones: PASAJERO y AGENCIA. También se vuelven a nombrar NUM_PASA-

Consulta: Dado el número de seguro social del instructor (\$X), encontrar los departamentos a los que pertenecen los cursos que él/ella ofrece

Especificación de navegación



Especificación SQL

```

    SELECT DEPARTAMENTO
    FROM CURSO, OFRECE
    WHERE CURSO.NOMBRE_CURSO = OFRECE.NOMBRE_CURSO AND NSS = $X
    o bien
    SELECT DEPARTAMENTO
    FROM CURSO
    WHERE NOMBRE_CURSO IN
    SELECT NOMBRE_CURSO
    FROM OFRECE
    WHERE NSS = $X
  
```

Figura 12.15. Especificación de navegación y SQL de una operación de recuperación de múltiples tipos de entidades.

JERO y NUM_AGENCIA en las relaciones PASAJERO y AGENCIA como un solo atributo, NUM_CLIENTE. Las dos interrelaciones de muchos a muchos TIENE_RESERVA y ESTA_A_BORDO corresponden de inmediato a relaciones nuevas. La figura 12.16 muestra el esquema relacional final; la clave primaria de cada relación está subrayada.

12.4.2. Correspondencia de operaciones con el modelo relacional

En el apartado 10.6 se introdujeron varios esquemas de navegación para la base de datos del caso de estudio. La figura 12.7 muestra correspondencias en SQL de los esquemas de navegación que se mostraron en las figuras 10.10 a 10.20. Las recuperaciones se expresan sobre todo a través de la construcción `SELECT...FROM...WHERE...`. Las inserciones se hacen a través de la construcción `INSERT INTO...VALUES...`. Las eliminaciones se hacen mediante `DELETE...FROM...WHERE...`, y las modificaciones a través de las construcciones `UPDATE...SET...WHERE...`. Los parámetros de entrada se muestran como variables

PASAJERO (NUM_CLIENTE, NOMBRE, TELEFONO, KILOMETRAJE, SITUACION)
 AGENCIA (NUM_CLIENTE, NOMBRE, TELEFONO, LIMITE_CREDITO)
 SEGMENTO_DE_RECORRIDO_DIARIO (NUMERO_VIAJE, NUMERO_SEGMENTO, FECHA,
 ASIENTOS_DISPONIBLES, ASIENTOS_RESERVADOS)
 SEGMENTO_DE_RECORRIDO_ORD (NUMERO_VIAJE, NUMERO_SEGMENTO, CIUDAD_SA-
 LIDA, HORA_SALIDA, CIUDAD_LLEGADA, HORA_LLEGADA, PRECIO, DISTANCIA)
 SEGMENTO_DE_RECORRIDO_POP (NUMERO_VIAJE, NUMERO_SEGMENTO, CIUDAD_SA-
 LIDA, HORA_SALIDA, CIUDAD_LLEGADA, HORA_LLEGADA, PRECIO, DISTANCIA)
 VIAJE (NUMERO_VIAJE, CIUDAD_SALIDA, CIUDAD_LLEGADA, DIAS_DE_SEMANA, TIPO,
 EVENTO)
 TIENE_RESERVA (NUM_CLIENTE, NUMERO_VIAJE, FECHA, NUMERO_SEGMENTO, NUM_A-
 SIENTO, OPCION_FUMAR)
 PASAJERO_ESTA_A_BORDO (NUM_CLIENTE, NUMERO_VIAJE, FECHA, NUMERO_SEG-
 MENTO)

Figura 12.16. El esquema relacional lógico del caso de estudio.

precedidas del signo \$. Las tres primeras operaciones (Fig. 12.17) están expresadas sobre la vista SEGMENTO_DE_RECORRIDO definida como la unión de SEGMENTO_DE_RECORRIDO_POP y SEGMENTO_DE_RECORRIDO_ORD. Tienen una estructura similar: un acceso a la vista SEGMENTO_DE_RECORRIDO basado en diferentes predicados de selección. En el primer caso la condición se expresa en términos de CIUDAD_SALIDA; en el segundo y tercer caso en términos de CIUDAD_SALIDA y CIUDAD_LLEGADA. Obsérvese que en la tercera operación se hace referencia a las distintas tuplas de SEGMENTO_DE_RECORRIDO que tienen el mismo NUMERO_VIAJE usando el recurso de *seudónimos*, con los seudónimos X e Y para la misma relación.

La operación O4 es una inserción simple; se expresa en dos versiones: una para PASAJERO y una para AGENCIA.

Para expresar la operación O5 se necesita incrustar la operación SQL en un lenguaje anfitrión, empleando una estructura de control «si ... entonces ... si no ...»: la reserva se hace sólo si el valor de ASIENTOS_DISPONIBLES es mayor que el número de asientos requeridos (expresado en la consulta con la variable \$ASIENTOS_REQ); si no, se genera un mensaje.

La operación O6 es una eliminación simple. La operación O7 es una eliminación compleja, donde el predicado de selección es una combinación de una condición según SITUACION y otra según la interrelación TIENE_RESERVA. Para expresar esta condición, se necesita usar la construcción de SQL NOT EXISTS.

Las operaciones O8 y O10 son simples actualizaciones, y la operación O9 es un SELECT sobre la relación pasajero. La operación O11 explora la relación TIENE_RESERVA para recuperar las reservas de pasajeros y agencias para un viaje en particular en una fecha dada; se muestran como consultas independientes. La operación O12 explora solamente la relación TIENE_RESERVA en conjunción con AGENCIA para recuperar todas las reservas de una agencia en particular.

12.5. Retroingeniería de los esquemas relationales a esquemas ER

En los últimos años, la popularidad del modelo relacional ha ido en aumento. Para muchas organizaciones, esto sucedió después que haber implantado bases de datos que usaban los sistemas de gestión de archivos y sistemas de bases de datos de los modelos que dominaban antes: el jerárquico y el de redes, y sus variantes según los distintos proveedores. Muchos DBMS no encajan en estos modelos comúnmente entendidos; por ejemplo, ADABAS (Software AG) o TOTAL (Cincom) usan sus versiones propias del modelo de redes. Como resultado de la proliferación de tales sistemas y de la variedad de formas en que éstos estructuran los datos, se ha hecho difícil para los analistas de bases de datos, diseñadores y administradores lograr un buen entendimiento conceptual de los datos que ya poseen bajo los distintos sistemas. Además se están diseñando bases de datos de una manera *ad hoc*; los diseñadores a menudo no utilizan la metodología de diseño conceptual seguido por diseño lógico que se ha recomendado en el curso de este libro. Es, por tanto, necesario emprender una actividad denominada **retroingeniería**, que aspira a extraer lo abstracto de lo concreto para obtener un entendimiento conceptual de las bases de datos existentes. Para tales abstracciones es apropiado usar un modelo de datos semántico más rico que el modelo implantado.

En este apartado se trata el problema de analizar y abstraer un conjunto dado de relaciones hasta obtener el esquema ER correspondiente. La representación objetivo de la base de datos en el modelo ER sería naturalmente más expresiva que la relacional; la información adicional que se requiere se deriva por lo regular de los conocimientos de los usuarios/diseñadores. El capítulo 13 contiene una explicación de la retroingeniería de bases de datos de redes y jerárquicas existentes. El capítulo 15, sobre herramientas de diseño, incluye la herramienta de retroingeniería de Bachman Information Systems. El procedimiento bosquejado a continuación se basa principalmente en el trabajo de Navathe y Awong mencionado en la bibliografía, aunque está más simplificado. Otros procedimientos basados en el concepto de dependencias de inclusión podrían considerarse más completos; sin embargo, lo que se busca aquí es presentar los conceptos básicos de la correspondencia inversa del modelo relacional a un modelo abstracto.

12.5.1. Suposiciones y tratamiento previo

Haremos la suposición implícita de que se nos da un conjunto de relaciones completamente normalizadas. Esto es, se espera que las relaciones estén en la forma normal Boyce-Codd (BCNF) o en la tercera forma normal (3NF), aunque podría haber algunas excepciones². En la práctica, podría ser necesario nor-

² Si se desea una descripción detallada de estas formas normales, véase Elmasri y Navathe (1989, Cap. 13). Obsérvese que la metodología también se aplica si las relaciones no están completamente normalizadas. Esto puede llevar a un esquema ER que necesite modificaciones posteriores.

Las operaciones O1, O2 y O3 se vuelven más complejas después de la decisión de dividir la entidad SEGMENTO_DE_RECORRIDO; sin embargo, el mecanismo de vistas de SQL permite volver a crear una relación virtual SEGMENTO_DE_RECORRIDO como la unión de SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP, como sigue:

```
CREATE VIEW SEGMENTO_DE_RECORRIDO (NUMERO_VIAJE, NUMERO_SEGMENTO,
FECHA, ASIENTOS_DISPONIBLES, ASIENTOS_RESERVADOS)AS
(SELECT*
  FROM SEGMENTO_DE_RECORRIDO_ORD)
(SELECT*
  FROM SEGMENTO_DE_RECORRIDO_POP)
```

Ahora se puede usar esta vista para las operaciones O1, O2 y O3

Operación O1: Buscar un segmento de recorrido por ciudad de salida (véase el esquema de navegación de la figura 10.10) \$C se refiere a la entrada de nombre de ciudad.

```
O1: SELECT*
  FROM SEGMENTO_DE_RECORRIDO
  WHERE CIUDAD_SALIDA = $C
```

Operación O2: Buscar un segmento de recorrido por ciudad de salida y ciudad de llegada (véase el esquema de navegación de la figura 10.11) \$S se refiere a la entrada de ciudad de salida y \$LL a la entrada de ciudad de llegada.

```
O2: SELECT*
  FROM SEGMENTO_DE_RECORRIDO
  WHERE CIUDAD_SALIDA = $S AND CIUDAD_LLEGADA = $LL
```

Operación O3: Buscar todos los viajes que tienen una parada intermedia en una ciudad determinada (véase el esquema de navegación de la figura 10.11). \$C se refiere a la entrada de nombre de ciudad.

```
O3: SELECT UNIQUE NUMERO_VIAJE
  FROM SEGMENTO_DE_RECORRIDO
  WHERE CIUDAD_LLEGADA = $C
  AND NUMERO_VIAJE IN
    SELECT NUMERO_VIAJE
    FROM SEGMENTO_DE_RECORRIDO
    WHERE CIUDAD_SALIDA = $C
```

Operación O4: Crear un cliente nuevo (véase el esquema de navegación de la figura 10.12). \$NUMC, \$NOMBRE, \$KILOMETRAJE, \$SITU son valores de entrada apropiados para un pasajero individual.

```
O4P: INSER INTO PASAJERO (NUM_CLIENTE, NOMBRE, TELEFONO, KILOMETRAJE, SITUACION)
  VALUES ($NUMC, $NOMBRE, $TELEFONO, $KILOMETRAJE, $SITU)
```

La operación de inserción para AGENCIA es como sigue: \$NUMC, \$NOMBRE, \$TELEFONO, \$CREDITO son valores de entrada apropiados para una agencia:

```
O4A: INSER INTO AGENCIA (NUM_CLIENTE, NOMBRE, TELEFONO, LIMITE_CREDITO)
  VALUES ($NUMC, $NOMBRE, $TELEFONO, $CREDITO)
```

Figura 12.17. Versiones en SQL de las operaciones del caso de estudio.

Operación O5: Hacer una reserva para un pasajero o agencia (véase el esquema de navegación de la figura 10.13). \$ASIENTOS_REQ (número de asientos requeridos), \$N (nombre del pasajero), \$C (número de cliente), \$PREF (preferencia de fumar-no fumar) se refieren a los valores introducidos en el momento de la búsqueda.

```
O5: SELECT ASIENTOS_DISPONIBLES INTO $DISP
      FROM SEGMENTO_DE_RECORRIDO_DIARIO
      WHERE FECHA = $F
            AND NUMERO_VIAJE = $V
            AND NUMERO_SEGMENTO = $S;
      si $DISP < $ASIENTOS_REQ entonces imprimir-mensaje («no hay suficientes asientos»)
      si-no comenzar
            INSERT INTO TIENE_RESERVA
            ($C, $V, $F, $S, $ASIENTOS_REQ, $PREF);
            UPDATE SEGMENTO_DE_RECORRIDO_DIARIO
            SET ASIENTOS_DISPONIBLES = ASIENTOS_DISPONIBLES - $ASIENTOS_REQ,
                ASIENTOS_RESERVADOS = ASIENTOS_RESERVADOS + $ASIENTOS_REQ
            WHERE FECHA = $F
                  AND NUMERO_VIAJE = $V
                  AND NUMERO_SEGMENTO = $S;
      terminar
```

Operación O6: Borrar reservas de un viaje pasado (véase el esquema de navegación de la figura 10.14). \$F, \$V, \$S tienen los valores correspondientes a una reserva pasada de un pasajero.

```
O6: DELETE TIENE_RESERVA
      WHERE FECHA = $F AND NUMERO_VIAJE = $V AND NUMERO_SEGMENTO = $S;
```

Operación O7: Borrar un cliente siempre que no sea «viajero frecuente» y no tenga una reserva (véase el esquema de navegación de la figura 10.15). \$N es el número de pasajero.

```
O7: DELETE FROM PASAJERO
      WHERE NUM_CLIENTE = $N AND SITUACION <> 'VIAJERO FRECUENTE'
      AND NOT EXISTS
            (SELECT*
              FROM TIENE_RESERVA
              WHERE PASAJERO.NUM_CLIENTE = $N);
```

Operación O8: Calificar a un cliente como viajero frecuente (véase el esquema de navegación de la figura 10.16). \$N es el número de pasajero.

```
O8: UPDATE PASAJERO
      SET KILOMETRAJE = 0
          SITUACION = 'VIAJERO FRECUENTE'
      WHERE NUM_CLIENTE = $N;
```

Operación O9: Recuperar la cantidad de kilómetros acumulados por los viajeros frecuentes (véase el esquema de navegación de la figura 10.17)

```
O9: SELECT NOMBRE, KILOMETRAJE
      FROM PASAJERO
      WHERE SITUACION= 'VIAJERO_FRECUENTE';
```

Figura 12.17.Bis Versiones en SQL de las operaciones del caso de estudio (continuación).

Operación O10: Actualizar el kilometraje de un viajero frecuente (véase el esquema de navegación de la figura 10.18). \$N contiene el número de un pasajero; \$K es el nuevo kilometraje.

```
O10: UPDATE PASAJERO
      SET KILOMETRAJE = $K
      WHERE NUM_CLIENTE = $N;
```

Operación O11: Recuperar todas las reservas vigentes de un viaje determinado (con número \$V) en una fecha determinada (igual a \$F) (véase el esquema de navegación de la figura 10.19).

```
O11: SELECT NOMBRE, TELEFONO, NUMERO_SEGMENTO, NUM_ASIENTO,
      OPCION_FUMAR
      FROM TIENE_RESERVA, PASAJERO
      WHERE FECHA=$F AND NUMERO_VIAJE =
      $V AND PASAJERO.NUM_CLIENTE=TIENE_RESERVA.NUM_CLIENTE;
```

Además, hacer lo siguiente para mostrar las reservas de agencias:

```
SELECT NOMBRE, TELEFONO, NUMERO_SEGMENTO, NUM_ASIENTO,
      OPCION_FUMAR
      FROM TIENE_RESERVA, AGENCIA
      WHERE FECHA=$F AND NUMERO_VIAJE=
      $V AND AGENCIA.NUM_CLIENTE=TIENE_RESERVA.NUM_CLIENTE
```

Operación O12: Recuperar todas las reservas vigentes de una agencia dada (véase el esquema de navegación de la figura 10.20). \$A contiene el nombre de la agencia.

```
O12: SELECT NUMERO_VIAJE, NUMERO_SEGMENTO, FECHA, NUM_ASIENTO,
      OPCION_FUMAR
      FROM TIENE_RESERVA, AGENCIA
      WHERE AGENCIA.NOMBRE=
      $A AND AGENCIA.NUM_CLIENTE=TIENE_RESERVA.NUM_CLIENTE
```

Figura 12.17.2Bis Versiones en SQL de las operaciones del caso de estudio (cont.).

malizar algunas relaciones antes de alcanzar 3NF o BCNF. Al escoger la forma normal alta se simplifica el proceso de correspondencia, pues implica relaciones que describen una entidad o interrelación cada una, y no muchas entidades juntas o una mezcla de entidades e interrelaciones. De hecho, ésta fue la motivación intuitiva de la normalización cuando Codd la propuso originalmente. No es razonable combinar información acerca de varias entidades en la misma relación excepto cuando se tenga que expresar una interrelación entre ellas (como se vio en los apartados 12.2.5 a 12.2.7).

Obsérvese que no es necesario imponer el requisito de BCNF o 3NF en un sentido muy estricto. Las consideraciones prácticas con frecuencia permiten almacenar una relación menos normalizada (es decir, en segunda o primera forma normal). Por ejemplo, considérese la relación

PERSONA (NSS, NOMBRE, CALLE, CIUDAD, ESTADO, CODIGO_POSTAL)

con la dependencia funcional CODIGO_POSTAL --- ESTADO. Está apenas en se-

gunda forma normal. En un diseño práctico de base de datos, no se crean anomalías de actualización al dar mantenimiento a esta relación porque no se espera referenciar y actualizar una relación (CODIGO_POSTAL, ESTADO) por sí misma. Por otro lado, descomponer la relación PERSONA en dos relaciones hace la recuperación de información de direcciones menos eficiente y requiere una reunión. Así pues, las relaciones en 2NF como ésta surgen por una de las siguientes razones, o por las dos: 1) no causan problemas en términos de actualización porque lógicamente no se justifica separar la información en múltiples relaciones; 2) pueden funcionar de manera más eficiente para los requisitos de aplicación especificados al minimizar el esfuerzo total de procesamiento.

El punto de inicio de nuestra metodología es un conjunto dado de relaciones en BCNF, 3NF y 2NF como las anteriores.

También haremos varias suposiciones acerca de los nombres de los atributos. Primero, se supone que a los atributos de clave primaria y de clave ajena en relaciones diferentes que tienen nombres idénticos (lo que es común) se les cambia el nombre, siempre que es necesario, de manera que se elimine la ambigüedad. Después del cambio de nombre, las restricciones referenciales (véase el apartado 12.1.1) deben resultar obvias. Un nombre de atributo puede construirse a partir de <nombrelrelación.nombreatributo> como una cadena colocando antes del nombre de atributo el nombre de la relación seguido por un punto. Las **dependencias de inclusión**, de la forma donde el conjunto de valores de un atributo A en una relación debe ser un subconjunto de los valores de algún atributo B en otra relación, *no requieren* un cambio de nombre de los atributos implicados. Por ejemplo, en estas relaciones:

PACIENTE (NSS, NOMBRE, NUM_HOSP, NUM_HABITACION)
 DOCTOR (NSS, NOMBRE, ESPECIALIDAD)
 PERSONA (NSS, NOMBRE)
 DOC_PAC (NSSD, NSSP)
 HOSPITAL (NUMERO_HOSP, DIRECCION, TAMAÑO)

se tienen las siguientes dependencias de inclusión:

NSS en PACIENTE	está incluido en	NSS en PERSONA
NSS en DOCTOR	está incluido en	NSS en PERSONA
NOMBRE en PACIENTE	está incluido en	NOMBRE en PERSONA
NOMBRE EN DOCTOR	está incluido en	NOMBRE en PERSONA

Estas no exigen ningún cambio de nombre; deben ser consideradas simplemente como información adicional que se usará más adelante en nuestra metodología. Sin embargo, se tienen las siguientes restricciones de integridad referencial:

NSSD en DOC_PAC	se refiere a	NSS en DOCTOR
NSSP en DOC_PAC	se refiere a	NSS en PACIENTE
NUM_HOSP en PACIENTE	se refiere a	NUMERO_HOSP en HOSPITAL

Para aclarar esta referencia, la mejor alternativa es volver a escribir la relación DOC_PAC como

DOC_PAC (NSS.DOCTOR, NSS.PACIENTE)

Otra posibilidad es volver a escribir las relaciones DOCTOR y PACIENTE como

PACIENTE (NSSP, NOMBRE, NUM_HOSP, NM_HABITACION)
DOCTOR (NSSD, NOMBRE, ESPECIALIDAD)

y no modificar DOC_PAC. El objetivo es hacer la equivalencia entre los atributos dentro de las relaciones tan explícita como sea posible. Se tendría que tomar nota de las dependencias de inclusión entre NSSD y NSS, y entre NSSP y NSS. No hay necesidad de cambiar el nombre de los sinónimos NUM_HOSP y NUMERO_HOSP, en tanto se tome nota del hecho de que representan el mismo atributo del mundo real.

Segundo, se supone que los atributos con el mismo nombre tienen un dominio idéntico (conjunto de valores) y la misma interpretación, sin importar dónde ocurran. Por ejemplo, en el conjunto de relaciones del ejemplo, los atributos NSS y NOMBRE caen en esta categoría. Sin embargo, pueden tener dependencias de inclusión entre ellos.

Si hay **homónimos**, esto es, atributos con nombres idénticos pero con significados distintos, se debe cambiar sus nombres. Por ejemplo, considérese

HOSPITAL (NOMBRE, NUMERO)
CURSO (NUMERO, NOMBRE)

En este caso, es obvio que NOMBRE en HOSPITAL y en CURSO tiene significados y dominios diferentes. También es posible que NUMERO en ambos sea un número entero; pero en el primero quizás describa el número de habitaciones, mientras que en el último puede representar un identificador asignado al curso. Por ello, un conjunto de nombres de atributos más significativo sería.

HOSPITAL (NOMBRE_HOSP, NUMERO_DE_HABITACIONES)
CURSO (NUMERO_CURSO, NOMBRE_CURSO)

Y así no buscaríamos equivocadamente relaciones entre esos atributos.

Nuestra tercera suposición es que ya se ha asignado claves primarias a las relaciones antes de que comience la correspondencia. También deben especificarse las claves candidatas para cada relación.

12.5.2. Una clasificación de las relaciones

A fin de poder realizar una abstracción de las relaciones a entidades e interrelaciones, se clasifican las relaciones como sigue:

1. **Relación primaria:** Relación cuya clave primaria no contiene otra clave de otra relación. Será convertida en entidad.
2. **Relación primaria débil:** Si la clave primaria CLP₁ de una relación R₁ contiene la clave primaria CLP₂ de una relación RD₂, la relación R₁ se denomina relación primaria débil. En este caso se dice que R₁ obtiene identificación externa de R₂. Considérese este ejemplo:

GASTOS_VIAJE (ID_VIAJE, FECHA, CODIGO_GASTO, IMPORTE)
VIAJE (ID_VIAJE, FECHA_INICIO, FECHA_FIN)

Aquí, la clave primaria de GASTOS_VIAJE incluye ID_VIAJE, que es la clave primaria de la relación VIAJE. Así, GASTOS_VIAJE es una relación primaria débil que obtiene identificación externa de VIAJE.

3. **Relación secundaria:** Relación cuya clave primaria está formada por una concatenación de claves primarias de otras relaciones. Una relación así surge cuando describe una interrelación entre aquellas relaciones cuyas claves están concatenadas. Por ejemplo, considérese las siguientes relaciones:

EMP_PROY (NUM_EMP, NUM_PROY, %ASIGNACION)
EMPLEADO (NUM_EMP, NOMBREE)
PROYECTO (NUM_PROY, NOMBREP)

Aquí, la relación EMP_PROY es una relación secundaria; describe en realidad una interrelación de EMPLEADO y PROYECTO al concatenar sus claves. Es posible concatenar de esta manera claves de más de dos relaciones para expresar interrelaciones de más alto grado.

Obsérvese que la clasificación anterior no es exhaustiva. Deja fuera algunos casos; por ejemplo, cuando la clave primaria de una relación es una concatenación no sólo de otras claves primarias, sino también de atributos adicionales. Nuestra intención es explicar las situaciones que ocurren más comúnmente, no pretender ser exhaustivos. Si desean un análisis más detallado, los lectores interesados pueden referirse a los trabajos de Navathe y Awong, Johannesson y Kalman, y Markowitz y Shoshani de la bibliografía.

12.5.3. Una metodología para establecer la correspondencia de un grupo de relaciones con un esquema ER

Se describirá el proceso de correspondencia como una serie de pautas o guías. Se aplican las suposiciones antes mencionadas.

Paso 1. Preprocesar y clasificar las relaciones. Las relaciones se someten a tratamiento previo, si es necesario, para satisfacer las suposiciones. Las relaciones se clasifican en las categorías descritas hasta donde sea posible antes de continuar el procedimiento.

Paso 2. Intercambiar las claves primarias y candidatas en ciertos casos.

El propósito de este paso es preparar las relaciones para los pasos siguientes que dependen de que se pueda detectar interrelaciones entre las claves primarias. Si la clave primaria de una relación R_1 coincide con la clave o claves candidatas de una o más relaciones (R_2 , R_3 , etc.), se designan esas claves candidatas de estas últimas como sus nuevas claves primarias. Por ejemplo, consideremos las siguientes relaciones de un sistema de cuentas por cobrar:

R_1 : CUENTA (NUM CUENTA, TIPO CUENTA, NOMBRE CUENTA)
 R_2 : CUENTA_PERSONAL (NSS, NUM CUENTA, NOMBRE PERSONA)
 R_3 : CUENTA_COMPAÑIA (NOMBRE COMPAÑIA, NUM CUENTA)

Obsérvese que NUM CUENTA es una clave candidata tanto de CUENTA_PERSONAL como de CUENTA_COMPAÑIA (suponiendo que cada cuenta pertenece a un individuo o bien a una compañía). Designar NUM CUENTA como clave primaria facilitaría la detección de la interrelación clase-subclase (o ES_UN) de R_2 y R_1 , y entre R_3 y R_1 . Por consiguiente, se vuelven a escribir las relaciones anteriores como

R_1 : CUENTA (NUM CUENTA, TIPO CUENTA, NOMBRE CUENTA)
 R_2 : CUENTA_PERSONAL (NUM CUENTA, NSS, NOMBRE PERSONA)
 R_3 : CUENTA_COMPAÑIA (NUM CUENTA, NOMBRE COMPAÑIA)

Paso 3. *Asignar nombres apropiados a partes de las claves que ocurren en relaciones secundarias para eliminar ambigüedades.* Este paso implica relaciones secundarias que se supone representan interrelaciones y relaciones apropiadas relacionadas con ellas, cuyas claves primarias están implicadas. Se ilustra esto en el siguiente ejemplo:

TRABAJADOR (NSS, TASA_HORA)
PILOTO (NSS, NUMERO_LICENCIA)
AVION (NUM REGISTRO, MODELO, AÑO)
REPARA (NSS, NUM REGISTRO, FECHA, HORAS)
VUELA (NSS, NUM REGISTRO, FECHA, HORAS)

Las relaciones REPARA y VUELA son secundarias, pero no está claro si NSS en REPARA y VUELA se refiere al número de seguro social de un piloto o de un trabajador. Esta ambigüedad se elimina al sustituir explícitamente el atributo NSS por su nombre completo apropiado en REPARA y VUELA:

REPARA (TRABAJADOR.NSS, NUM REGISTRO, FECHA, HORAS)
VUELA (PILOTO.NSS, NUM REGISTRO, FECHA, HORAS)

Paso 4. *Hacer corresponder relaciones primarias con entidades.* En esta etapa, ya se ha hecho el trabajo preparatorio para detectar diferentes tipos de interrelaciones entre las relaciones. La existencia de una relación primaria sugiere la presencia de una entidad en el nivel conceptual. Por tanto, se define una

entidad por cada relación primaria. Se supone que el nombre por defecto de la entidad puede ser el mismo que el nombre de la relación, aunque el diseñador puede cambiar el nombre. Esto se da por supuesto en los pasos subsecuentes de la correspondencia.

Paso 5. *Hacer corresponder las relaciones primarias débiles con entidades identificadas externamente.* La existencia de una relación primaria débil sugiere la presencia de una entidad en el nivel conceptual que sea identificada externamente por la entidad correspondiente (definida en el paso 4) y cuya clave se importe de ella. Por tanto, se define una entidad que corresponde a la relación primaria débil y se indica que tiene identificación externa.

Paso 6. *Detectar interrelaciones de subconjunto entre las entidades.* En el paso 2 se dijo que puede requerirse cierto tratamiento previo para detectar relaciones que tienen la misma clave primaria, y en el paso 4 estas relaciones se hicieron corresponder con entidades. *Sobre la base del conocimiento externo de estas entidades*, el diseñador puede ahora especificar manualmente las interrelaciones de subconjunto entre estas clases de entidades. Por ejemplo, de las tres relaciones utilizadas en el paso 2, puede resultar una jerarquía de generalización, con CUENTA como superclase y CUENTA_PERSONAL y CUENTA_COMPAÑIA como clases subconjunto. Las designaciones total/parcial y exclusiva/superpuesta de la generalización también se especifican *con base en el conocimiento del diseñador*. En el caso anterior, por ejemplo, se puede asignar (t,e) a esta generalización para designar que las subclases son totales y exclusivas.

Paso 7. *Hacer corresponder relaciones secundarias con interrelaciones de entidades.* La existencia de una relación secundaria sugiere la presencia de una interrelación entre las entidades correspondientes en el nivel conceptual (definidas en el paso 4). La designación de las entidades apropiadas se facilita por el tratamiento del paso 3. Cada relación secundaria corresponde a una interrelación entre las entidades cuyas claves se concatenan para formar su clave primaria. Obsérvese que *no hay manera de asignar automáticamente* las restricciones de cardinalidad mínima y máxima de las entidades participantes respecto a la interrelación recién creada. El diseñador puede ahora especificar manualmente las restricciones de cardinalidad.

En el caso de las relaciones secundarias

REPARA (TRABAJADOR.NSS, NUM_REGISTRO, FECHA, HORAS)
VUELA (PILOTO.NSS, NUM_REGISTRO, FECHA, HORAS)

generadas al final del paso 3, se definen las interrelaciones REPARA entre las entidades TRABAJADOR y AVION, y VUELA entre PILOTO y AVION. La correspondencia de este ejemplo se muestra en la figura 12.18, con una selección de car-

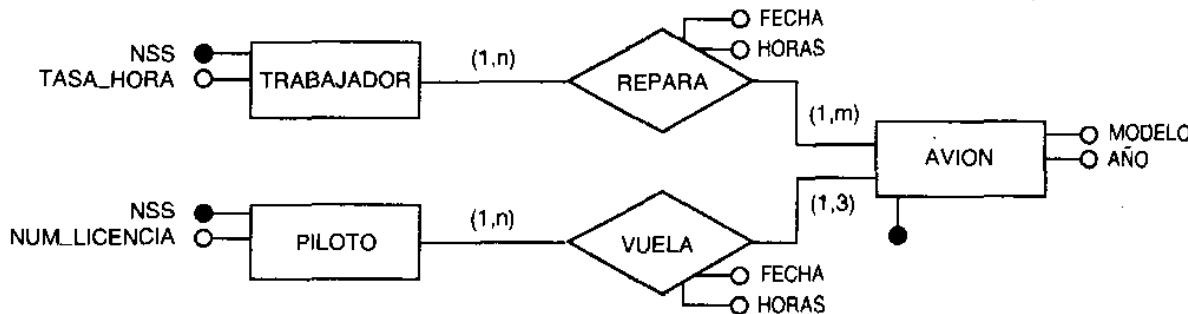


Figura 12.18. Correspondencia de relaciones secundarias.

dinalidades suministradas normalmente por el diseñador³. Si las claves de tres relaciones o más están involucradas, el resultado es una relación n-aria.

Paso 8. *Hacer corresponder las restricciones de integridad referencial de los atributos que no son claves con interrelaciones de entidades.* La presencia de un atributo de clave ajena en un relación, que indica su restricción referencial respecto a la clave primaria de otra relación, significa que hay una interrelación entre las dos entidades correspondientes. Esta interrelación es diferente de la indicada por las relaciones secundarias. Hay dos diferencias:

1. Las relaciones secundarias pueden tener atributos; éstos se transforman en atributos de la interrelación. La interrelación modelada a través de un atributo de clave ajena no tiene atributos.
2. La interrelación que corresponde a una relación secundaria es normalmente una interrelación de muchos a muchos; si fuera de otra manera, la clave no sería mínima. La interrelación modelada a través de un atributo de clave ajena, en cambio, no puede ser de muchos a muchos, porque la clave ajena apunta a un solo valor de la clave primaria. Puede ser o de uno a uno o de uno a muchos.

La restricción referencial indicada por el atributo de clave ajena (que no es clave) corresponde con una interrelación aparte. Se ilustra con un ejemplo.

EMPLEADO (NSS, NOMBRE, DIRECCION, NUM_DEPT, NSSDIR)
DEPARTAMENTO (NMD, NOMBRED, UBICACION)

El atributo NUM_DEPT es una clave ajena; tiene una restricción referencial respecto a NMD en DEPARTAMENTO. Este corresponde con una interrelación entre las entidades EMPLEADO y DEPARTAMENTO, como se muestra en la figura

³ Obsérvese que la razón de cardinalidad máxima entre, digamos, TRABAJADOR y AVION, debe ser de muchos a muchos. Si fuera de otra manera (por ejemplo, si un trabajador puede reparar solamente un avión), la clave primaria de REPARA sería sólo TRABAJADOR.NSS.

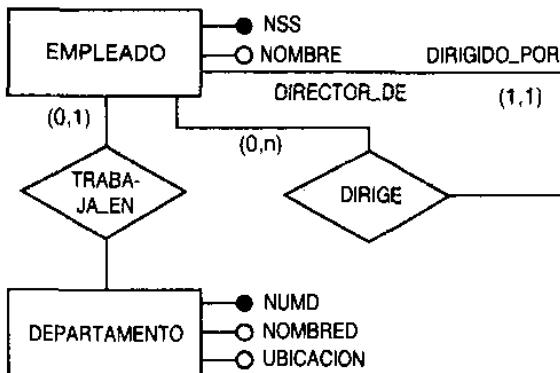


Figura 12.19. Correspondencia de restricciones referenciales.

12.19. El atributo NUM_DEPT puede aparecer o no en la entidad resultante EMPLEADO. El atributo NSSDIR tiene una restricción referencial respecto a la clave primaria NSS de la misma relación. Corresponde con la relación cíclica DIRIGE, como se muestra en la figura. La entidad EMPLEADO desempeña dos papeles en esta interrelación: DIRECTOR_DE y DIRIGIDO_POR, que serían suministrados por el diseñador.

Paso 9. *Hacer corresponder relaciones no clasificadas.* Hasta ahora se ha cubierto una gran parte de las situaciones que ocurren comúnmente entre las relaciones y se ha mostrado paso por paso como pueden hacerse corresponder con entidades e interrelaciones. El procedimiento fue guiado por la detección de interrelaciones, que se descubren al observar los atributos de claves primarias y candidatas, las restricciones referenciales, y las dependencias de inclusión. Obsérvese que se han ignorado algunas combinaciones peculiares de claves y no claves que ocurren de vez en cuando. Estas se tienen que resolver caso por caso. Por ejemplo, consideremos las siguientes relaciones:

DEPARTAMENTO (NUMD, NOMBRED, NOMBRE_DIR)
UBICACIONES_DEPTOS (NUMD, UBIC)

Aquí la relación DEPARTAMENTO es primaria, pero UBICACIONES_DEPTOS no es una relación primaria débil porque no puede ser considerada como una ubicación que DEPARTAMENTO identifica externamente. Además, no es secundaria porque UBIC no es una clave primaria de otra relación. Hay dos alternativas: 1) hacer UBIC un atributo polivalente de la entidad DEPARTAMENTO, o 2) «elevar» UBIC a una entidad con el atributo UBIC y definir una interrelación entre esta entidad y DEPARTAMENTO. La primera alternativa parece preferible.

12.5.4. Ejemplo de retroingeniería

En esta sección se presenta un ejemplo completo de correspondencia del modelo relacional al modelo ER, a fin de ilustrar el procedimiento anterior. Con-

sidérese una base de datos de un aeropuerto privado que da servicio a aviones pequeños. Contiene el siguiente conjunto de relaciones relativas a los aviones, propietarios de aviones (que pueden ser individuos o compañías), pilotos, empleados que atienden aviones, etcétera.

COMPAÑIA (NOMBRE, DIRECCION, TELEFONO)
PERSONA (NSS, NOMBRE, DIRECCION, TELEFONO)
PROPIETARIO (NSS, NOMBRE, DIRECCION, TELEFONO)
TRABAJADOR (NSS, NOMBRE, DIRECCION, TELEFONO, SALARIO, TURNO)
PILOTO (NSS, NUM_LICENCIA, NOMBRE, DIRECCION, TELEFONO, RESTRICCIONES)
OFICIAL_COMPAÑIA (NOMBREC, PUESTO, NOMBRE, NSS, NUM_REG)
AVION (NUM_REGISTRO, MODELO, TIPO)
PERSONA_POSEE (NSS, NUM_REGISTRO FECHA)
COMPAÑIA_POSEE (NOMBRE, NUM_REGISTRO, FECHA)
VUELA (NSS, NUM_REGISTRO)
MANTIENE (NSS, NUM_REGISTRO)

Se tienen las siguientes dependencias de inclusión:

NSS en PROPIETARIO	está incluido en	NSS en PERSONA
NSS en TRABAJADOR	está incluido en	NSS en PERSONA
NOMBRE en PILOTO	está incluido en	NOMBRE en PERSONA

Estas no exigen cambios de nombre. Se tienen también las siguientes restricciones de integridad referencial:

NSS en PERSONA_POSEE	se refiere a	NSS en PROPIETARIO
NSS en VUELA	se refiere a	NSS en PILOTO
NSS en MANTIENE	se refiere a	NSS en TRABAJADOR
NUM_REGISTRO, en PERSONA_POSEE COMPAÑIA_POSEE, VUELA, MANTIENE	se refieren a	NUM_REGISTRO en AVION
NUM_REG en OFICIAL_COMPAÑIA	se refiere a	NUM_REGISTRO en AVION

Obsérvese que NUM_REG en OFICIAL_COMPAÑIA puede tener valores nulos para aquellos oficiales de la compañía que no poseen aviones. Adviértase también que NSS es una clave candidata en OFICIAL_COMPAÑIA. Examinemos la correspondencia paso por paso.

Paso 1. Se cambia el nombre del atributo NOMBRE en COMPAÑIA y COMPAÑIA_POSEE, donde representa el nombre de una compañía, a NOMBREC a fin de que coincida con el atributo correspondiente en OFICIAL_COMPAÑIA. En seguida se clasifican las relaciones con base en sus definiciones, como sigue:

COMPAÑIA (NOMBREC, DIRECCION, TELEFONO): **PRIMARIA**
 PERSONA (NSS, NOMBRE, DIRECCION, TELEFONO): **PRIMARIA**
 PROPIETARIO (NSS, NOMBRE, DIRECCION, TELEFONO): **PRIMARIA**
 TRABAJADOR (NSS, NOMBRE, DIRECCION, TELEFONO, SALARIO, TURNO):
PRIMARIA
 PILOTO (NSS, NUM_LICENCIA, NOMBRE, DIRECCION, TELEFONO, RESTRIC-
 CIONES): **PRIMARIA**
 OFICIAL_COMPAÑIA (NOMBREC, PUESTO, NOMBRE, NSS, NUM_REG) **PRIMA-
 RIA DEBIL**
 AVION (NUM_REGISTRO, MODELO, TIPO): **PRIMARIA**
 PERSONA_POSEE (NSS, NUM_REGISTRO, FECHA): **SECUNDARIA**
 COMPAÑIA_POSEE (NOMBREC, NUM_REGISTRO, FECHA): **SECUNDARIA**
 VUELA (NSS, NUM_REGISTRO): **SECUNDARIA**
 MANTIENE (NSS, NUM_REGISTRO): **SECUNDARIA**

Paso 2. Tenemos la opción de hacer a NSS la clave primaria de OFICIAL_COMPAÑIA. Sin embargo, suponemos que en general se hace referencia a los oficiales de las compañías en términos de la compañía, y escogemos mantener la situación de OFICIAL_COMPAÑIA como primaria débil, dependiente de la compañía para su identificación. Este es un ejemplo de decisión subjetiva del diseñador. El esquema ER final resulta afectado por tales decisiones.

Paso 3. Las claves de PERSONA_POSEE, VUELA y MANTIENE contienen NSS como atributo. Para poder saber sin ambigüedad a que NSS se refieren, cambiamos sus nombres a PROPIETARIO.NSS, PILOTO.NSS y TRABAJADOR.NSS.

Paso 4. Las relaciones primarias COMPAÑIA, PERSONA, PROPIETARIO, TRABAJADOR, PILOTO y AVION se transforman en entidades con los atributos correspondientes. Por defecto, se utilizan los mismos nombres.

Paso 5. La relación OFICIAL_COMPAÑIA corresponde con una entidad que se identifica externamente (con NOMBREC como identificador externo) mediante la entidad COMPAÑIA.

Paso 6. Se observa la clave primaria idéntica en las relaciones PERSONA, PROPIETARIO, TRABAJADOR y PILOTO. Con nuestro conocimiento de las interrelaciones ES_UN, creamos una jerarquía de generalización⁴, con PERSONA como superclase de la jerarquía, eligiendo parcial y superpuesta como tipo de la generalización.

Paso 7. Las relaciones secundarias PERSONA_POSEE, COMPAÑIA_POSEE, VUELA y MANTIENE se convierten en interrelaciones entre entidades apropiadas. Las restricciones de cardinalidad asignadas están basadas en nuestras suposiciones acerca de aplicación.

⁴ Si se eligiera NSS como clave de OFICIAL_COMPAÑIA, también se habría modelado OFICIAL_COMPAÑIA como subconjunto de PROPIETARIO en esta jerarquía.

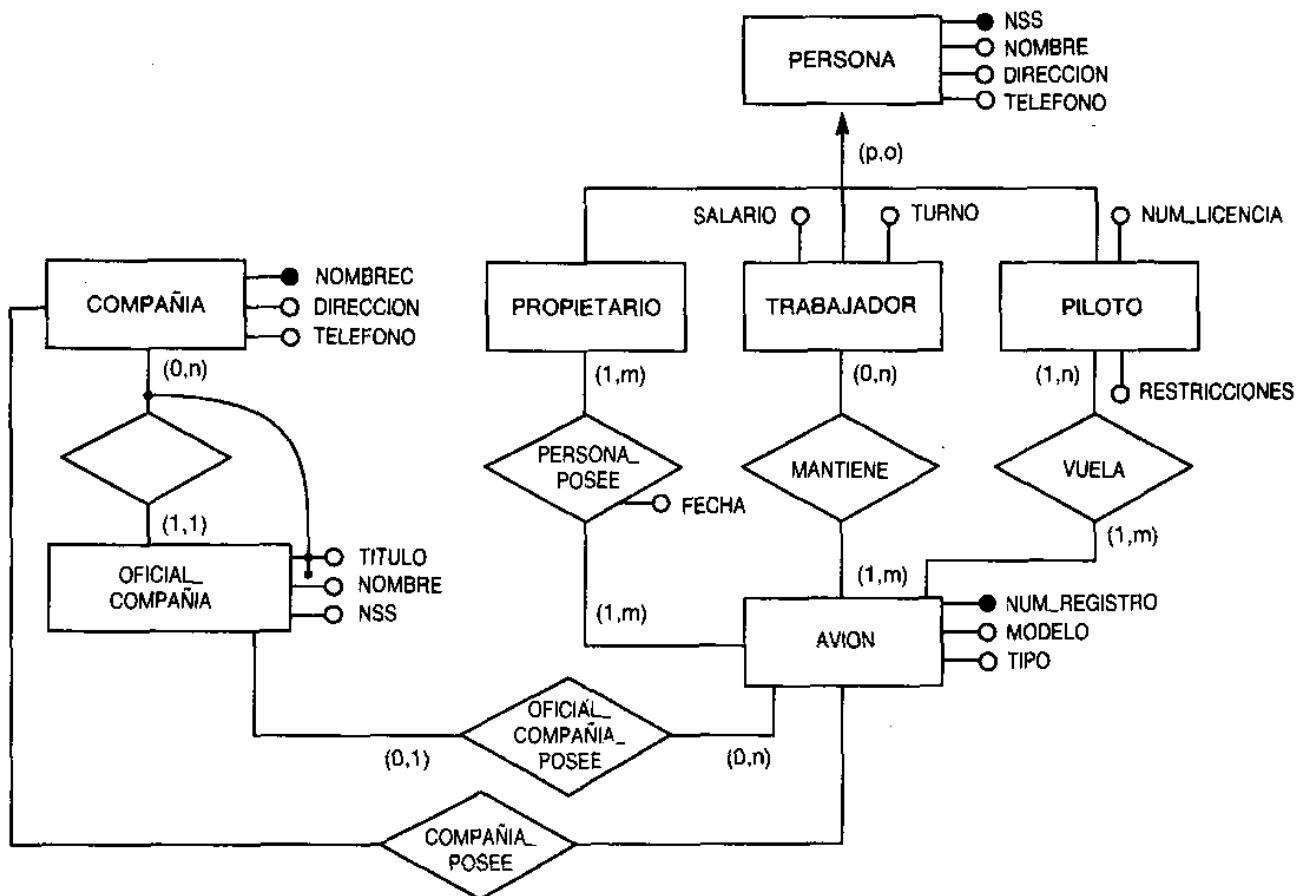


Figura 12.20. Esquema ER definido a partir de relaciones en el ejemplo del aeropuerto.

Paso 8. Este se encarga de las restricciones referenciales de NUM_REG en OFICIAL_COMPAÑIA respecto a la clave primaria de AVION. Se convierte en la interrelación OFICIAL_COMPAÑIA_POSEE. Se asigna una cardinalidad mínima de cero del lado de OFICIAL_COMPAÑIA de la interrelación para indicar que no todos los oficiales de la compañía poseen un avión.

Paso 9. Ya se ha señalado que NSS podría ser considerado como clave primaria de OFICIAL_COMPAÑIA. Por tanto, esta entidad podría colocarse como subconjunto de PERSONA o bien de PROPIETARIO en la jerarquía de generalización antes definida. Hacer eso sería suponer que los oficiales de la compañía tienen casos en las relaciones PERSONA o bien PERSONA y PROPIETARIO, respectivamente, en la base de datos dada. No se incluyen estas alternativas en la figura 12.20, que muestra el resultado final de la correspondencia.

12.6. Resumen

La estructura del modelo relacional es bastante simple; se define un esquema como una colección de relaciones, cada una con una clave primaria. En este capítulo presentamos los conceptos básicos del modelo de datos relacional, incluidos los conceptos de dominio, atributos y claves candidatas y primarias. Analizamos tres tipos de restricciones de integridad: restricciones de claves, de integridad de entidades y de integridad referencial. La integridad referencial implica la noción de clave ajena. Se mencionaron las formas normales de las relaciones, pero no se expusieron en detalle.

A continuación se analizó la correspondencia de esquemas ER después de haber sido procesados como se explicó en el capítulo 11. Esto implicó resolver los identificadores externos así como los atributos compuestos y polivalentes antes de establecer la correspondencia de las entidades con las relaciones. La correspondencia de las interrelaciones se explicó individualmente para las interrelaciones de uno a uno, de uno a muchos y de muchos a muchos, y para las interrelaciones n-arias. El esquema ER final del caso de estudio de los capítulos 10 y 11 se hizo corresponder con un conjunto de relaciones usando el procedimiento anterior.

Como se aplicó normalización a las entidades o interrelaciones (Cap. 6), las relaciones resultantes están automáticamente en una forma normal deseada. Se puede verificar la normalización en el esquema relacional final, o tal vez aplicarle normalizaciones adicionales. El diseño lógico de una base de datos relacional a partir de un esquema conceptual ER implica no sólo la correspondencia de los datos sino también de las operaciones y consultas. Se presentó un procedimiento general para traducir consultas de «selección-proyección-reunión» de sus esquemas de navegación a SQL, que es actualmente el lenguaje estándar *de facto* para el modelo relacional. Se mostró la correspondencia de los doce esquemas de navegación del capítulo 10 con SQL para el caso de estudio.

Finalmente se trató el problema de la retroingeniería desde un conjunto dado de relaciones hasta un esquema conceptual abstracto en el modelo ER. Este es un ejercicio muy útil en organizaciones en las que las bases de datos se han acumulado durante largos períodos o han sido convertidas en sistemas relacionales sin un diseño adecuado de base de datos. En la mayoría de las organizaciones, un entendimiento conceptual de lo que ya existe puede ser muy útil. Presentamos un procedimiento para clasificar primero las relaciones en diferentes tipos con base en las interrelaciones de claves primarias dentro de las relaciones. Luego, usando esta clasificación, cambiando los nombres de los atributos, etc., describimos un procedimiento paso por paso para llegar al esquema conceptual. Para suministrar la expresividad adicional presente en el esquema ER, el procedimiento tiene que contar con información adicional, como las interrelaciones conjunto-subconjunto o las restricciones de cardinalidad provistas por el diseñador.

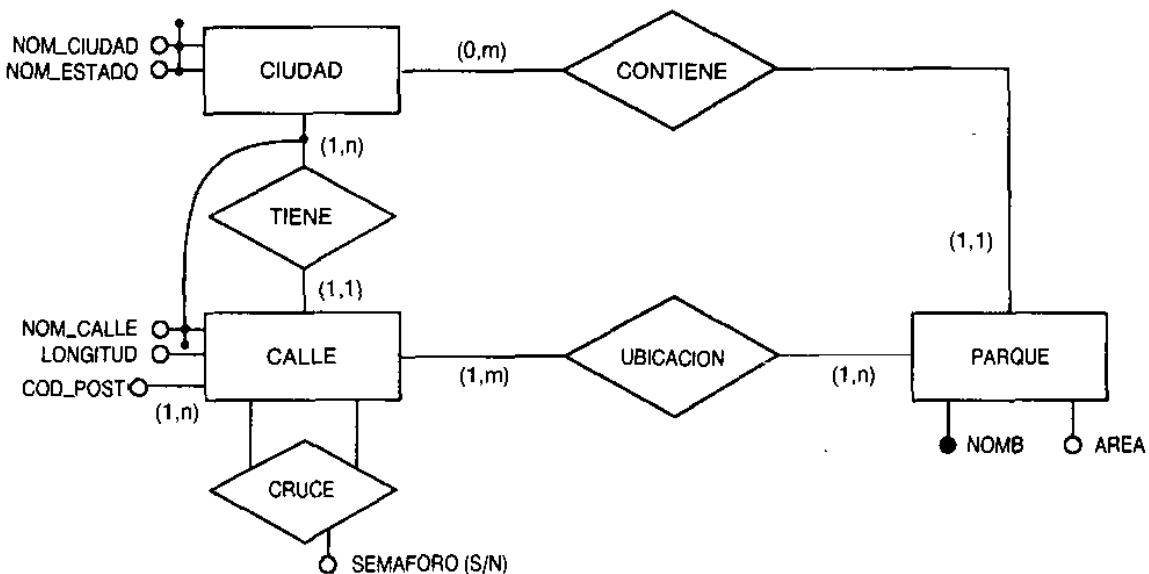


Figura 12.21. Esquema ER de una base de datos para los departamentos de tráfico y de parques de una ciudad.

Ejercicios

- 12.1. Considere la base de datos de proyectos de investigación de la figura 2.35. Aplique la metodología descrita en este capítulo para transformar el esquema lógico ER en un esquema relacional.
- 12.2. Considere la base de datos universitaria de la figura 2.33. Reemplace los conjuntos-subconjuntos introduciendo interrelaciones entre las entidades genéricas y subconjunto. Aplique la metodología descrita en este capítulo para transformar el esquema lógico ER en un esquema relacional.
- 12.3. Repita el ejercicio 12.2 con la base de datos de fútbol de la figura 2.34.
- 12.4. Considere el diagrama ER de la figura 12.21 que representa una base de datos para los departamentos de tráfico y parques de una ciudad. Dibuje esquemas de navegación en forma ER de las siguientes consultas:
 1. Listar todos los nombres de las calles de una ciudad dada.
 2. Listar todas las intersecciones de la calle Main de la ciudad de Gainesville, Georgia.
 3. Listar todos los parques de la ciudad de Gainesville, Florida.
 4. Listar todos los parques localizados en la calle Huron de Ann Arbor, Michigan.

Proceda como sigue:

- a) Primero convierta el modelo ER en una base de datos relacional
- b) Convierta los esquemas de navegación mencionados a SQL.

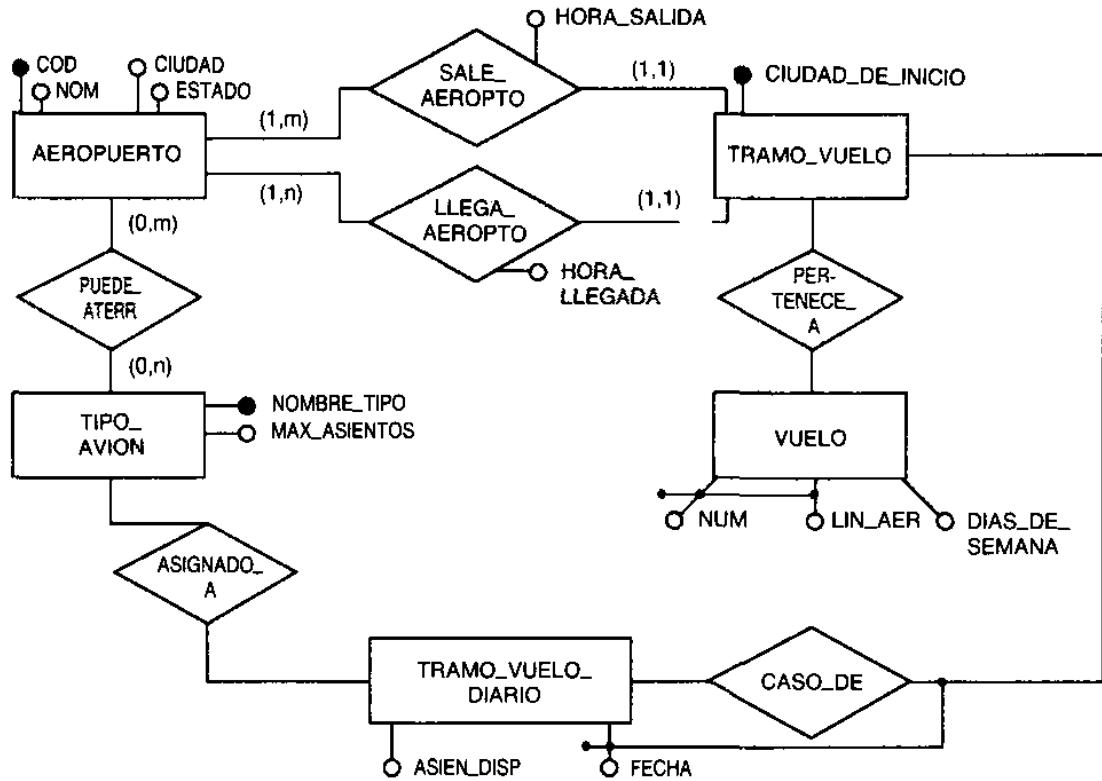


Figura 12.22. Esquema ER de una base de datos de aeropuerto y vuelos.

- 12.5. Considere el esquema ER de la base de datos de aeropuerto y vuelos de la figura 12.22. Conviértalo en un conjunto de relaciones. Señale todas las restricciones referenciales entre las relaciones.
- 12.6. Considere el esquema ER al que se llegó en la figura 12.20. Convierta la jerarquía de generalización introduciendo interrelaciones ES_UN. Con su conocimiento de este ejemplo, aplique el procedimiento de correspondencia de ER a relacional del apartado 12.2 y conviértalo en un conjunto de relaciones. Verifique si sus resultados son iguales al esquema con el que iniciamos el apartado 12.5.4. Determine las razones de cualesquier discrepancias.
- 12.7. Considere las siguientes relaciones de la base de datos de una biblioteca:

LIBRO (NUM_CATALOGO, NOMBRE)
ESCRIBIR (NUM_CATALOGO, NOMBRE_AUTOR, AÑO, NUM_EDICION)
AUTOR (NOMBRE_AUTOR, DIRECCION, NOMBRE_UNIVERSIDAD)
CONTRIBUCION (NUM_CATALOGO, NOMBRE_AUTOR, NUM_CAPITULO)
LIBRO_DE_TEXTO (NOMBRE_UNIVERSIDAD, NUM_CURSO, NUM_CATALOGO)
MANUSCRITO (NUM_CATALOGO, ASUNTO)
UNIVERSIDAD (NOMBRE_UNIVERSIDAD, CIUDAD)

Las relaciones no requieren explicación; cláifiquelas como se sugiere en el apartado 12.5. Identifique las dependencias de inclusión y restricciones referenciales; luego conviértalas en un esquema ER. Observe que

los libros de textos y manuscritos son casos especiales de libros y que no todos los autores pertenecen necesariamente a universidades.

- 12.8. Considere el siguiente conjunto de relaciones mantenidas por el departamento de personal de una compañía:

EMPLEADO (NOMBRE, INICIAL, APELLIDO, NSS, FECHANAC DIRECCION, SEXO, SALARIO, NSSSUPER, NMD)
DEPARTAMENTO (NUMEROD, NOMBRED, NSSDIR)
PROYECTO (NUMEROP, NOMBREP, NUMD)
DEPENDIENTE (NSSEMP, NOMBRE_DEP, SEXO, FECHANAC, RELACION)
UBICACIONES_DEPTOS (NUMD, LUGD, NUM_EMPL)
TRABAJA_EN (NSSEMP, NUMP, HORAS)

El significado de las relaciones es obvio. NUMD y NUMEROD representan el número de departamento; NUMP y NUMEROP se refieren a los números de los proyectos; NSS es una clave candidata en EMPLEADO; NSSEMP también representa el número de seguro social del empleado. Muestre las restricciones referenciales en esta base de datos; luego, siga la metodología de este capítulo para hacer corresponder las relaciones anteriores con un esquema ER. Seleccione las restricciones de cardinalidad apropiadas y justifique sus selecciones.

- 12.9. Partiendo de las relaciones que se muestran en la figura 12.16 y con su conocimiento del caso de estudio, aplique el procedimiento de correspondencia inversa para convertirlas en un esquema ER. Compare el resultado final con el esquema ER al que se llegó en el capítulo 11.

Bibliografía

E.F. Codd, «A Relational Model of Data for Large Shared Data Banks», *Communications of the ACM*, 13, 1970, págs. 377-87.

Este es el trabajo clásico sobre el modelo de datos relacional. La idea que inspiró a Codd fue desarrollar un modelo de simplicidad espartana en el cual los aspectos lógicos se distinguieran claramente de los rasgos orientados a la implantación.

Si se desea estudiar los rasgos básicos del modelo relacional y del lenguaje SQL, véanse los libros de Elmasri y Navathe, Date, Korth y Silberschatz, y Ullman listados en la bibliografía del capítulo 1. Elmasri y Navathe explican las correspondencias del modelo ER y de una extensión del modelo ER con todos los modelos de manera independiente. Analizan exhaustivamente las formas normales y tratan la normalización con y sin la declaración de claves primarias.

E.F. Codd, *The Relational Model for Database Management-Version 2*, Addison-Wesley, 1990.

Este es un resumen de 333 rasgos que pueden ser descritos como pertinentes, útiles y deseables para el modelo de datos relacional. De éstos, cerca de 55 estuvieron ya presentes en el modelo de datos relacional original. Los rasgos están divididos en categorías que incluyen tipos de datos, autorizaciones, catálogos, restricciones de integridad, nombres,

vistas, bases de datos distribuidas, y así sucesivamente. Las motivaciones de la versión 2 del modelo de datos son: 1) mantener todos los rasgos de la versión 1; 2) resolver los errores y omisiones cometidos en la implantación del modelo original; 3) ampliar el ámbito del modelo relacional a la gestión generalizada de bases de datos; 4) añadir nuevos recursos respecto a reuniones, actualización de vistas, gestión de bases de datos distribuidas, etc., y 5) indicarles a los usuarios lo que se están perdiendo con los productos relacionales presentes.

J. Schmidt y M. Brodie, *Relational Database Systems: Analysis and Comparison*. Springer-Verlag, 1983.

Varios sistemas de gestión de bases de datos relacionales son revisados y comparados con base en un catálogo de rasgos desarrollado por el Relational Database Task Group (grupo de trabajo sobre bases de datos relacionales). La comparación se hace en términos de los siguientes rasgos: componentes de la base de datos, capacidades funcionales, definiciones de esquemas, recursos de generación y administración, clases funcionales, interfaces, arquitecturas de sistemas y aspectos operativos.

P. Bertaina, A. di Leva y P. Giolito, «Logical Design in CODASYL and Relational Environments». En S. Ceri, ed., *Methodology and Tools for Database Design*, North-Holland, 1983.

T. Teorey, D. Yang y J. Fry, «A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model». *ACM Computer Surveys*, 18, núm. 2, julio de 1986.

E. Wong y R. Katz, «Logical Design and Schema Conversion for Relational and DBTG Databases». En P. Chen, ed., *Entity-Relationship Approach to System Analysis and Design*, North-Holland, 1980, 311-22.

Estos trabajos dan reglas generales para la transformación de un esquema ER en un esquema relacional. En el trabajo de Teorey, Yang y Fry se consideran varios casos que tienen en cuenta valores nulos. De acuerdo con el valor de cardinalidad mínima (0 ó 1), se expresan restricciones de integridad apropiadas en el esquema relacional para permitir o prohibir los valores nulos en las relaciones resultantes.

H.A. Schmid y J.R. Swenson, «On the Semantics of the Relational Data Model», *Proc. 1975 ACM-SIGMOD Conference on the Management of Data*. Disponible de ACM.

Este trabajo viejo pero importante es una tentativa de investigar las formas normales con base en una interpretación semántica. Diferentes tipos de relaciones y sus 3NF correspondientes se categorizan con base en sus significados al representar la realidad. Este trabajo pionero también indica que los modelos conceptuales producen esquemas normalizados de forma natural.

S.R. Dumpala y S. K. Arora, «Schema Translation Using the Entity-Relationship Approach». En P. Chen, ed., *Entity-Relationship Approach to Information Modeling and Analysis*, North-Holland, 1983.

Esta es la primera referencia que considera todos los casos de correspondencias: ER a relacional, de redes y jerárquica, así como la correspondencia inversa de relacional, de redes y jerárquica a ER. El planteamiento está limitado al modelo ER básico.

H. Briand, H. Habrias, J.F. Hue y Y. Simon, «Expert System for Translating an ER Diagram into Databases». En J. Lin, ed., *Proc. Fourth International Conference on Entity-Relationship Approach, Chicago*, IEEE Computer Society, 1985.

Este trabajo desarrolla un enfoque de los diferentes casos de correspondencia (ER a diagramas Bachman y ER a bases de datos relacionales) construyendo primero un metaesquema que representa el diagrama ER en una red semántica. El enfoque representa el metaesquema en predicados PROLOG y aplica diferentes reglas de transformaciones en un sistema experto basado en reglas.

S.B. Navathe y A.M. Wong, «Abstracting Relational and Hierarchical Data with a Semantic Data Model». En S. March, ed., *Proc. Sixth International Conference on Entity-Relationship Approach*, North-Holland, 1987.

Este trabajo trata el problema de correspondencia inversa de una base de datos jerárquica o relacional con un esquema ER extendido, con categorías para apoyar las generalizaciones y las interrelaciones conjunto-subconjunto. Amplia el trabajo de Dumpala y Arora. Las relaciones se clasifican teniendo en cuenta los papeles desempeñados por las claves primarias, candidatas y ajena. Usando la clasificación de relaciones, el artículo da un procedimiento paso por paso para la correspondencia inversa.

P. Johannesson y K. Kalman, «A Method for Translating Relational Schemas into Conceptual Schemas». En C. Batini, ed., *Proc. Seventh International Conference on the Entity-Relationship Approach*, North-Holland, 1988, 279-94.

Este artículo es una ampliación del trabajo ya mencionado de Navathe y Awong, pero está restringido al modelo relacional. Su método para la retroingeniería de una base de datos relacional usa dependencias de inclusión; es un método más general que el enfoque de Navathe y Awong basado en la clasificación de las relaciones. Los autores han llevado a la práctica una herramienta de sistema experto basada en este método.

V. M. Markowitz y A. Shoshani, «On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model», *Proc. ACM-SIGMOD Conference on Management of Data*, 1989, 430-39.

V. M. Markowitz y A. Shoshani, «Name Assignment Techniques for Relational Schemas Representing Extended Entity-Relationship Schemas». En F. Lochovsky, ed., *Proc. Eighth International Conference on Entity-Relationship Approach*, North-Holland, 1989.

Estos dos trabajos examinan los temas de asignación de nombres y corrección a la luz de los esquemas ER y relacionales equivalentes. Los autores intentan demostrar que la capacidad de información de esquemas equivalentes bajo su sistema de correspondencia es en realidad la misma.

T. Teorey y D. Yang, «Usage Refinement for ER-to-Relation Design Transformations». Por aparecer en *Information Sciences, an International Journal*, 1990.

K. H. Davis, «Need for "Flexibility" in a Conceptual Model», *Information and Management*, 18, septiembre de 1990, 231-41.

Este trabajo trata la cuestión del comportamiento dinámico y cómo afecta los modelos de datos producidos. Se comparan dos ejemplos de retroingeniería para la transformación del modelo relacional al modelo ER.

T. Johnston, «Building Logical Data Models», *Computerworld*, 4 de abril de 1985.

Este artículo describe un enfoque ascendente para obtener un esquema relacional completamente normalizado a partir de las estructuras de archivos existentes en un sistema dado. De esta manera pasa por alto la conversión en esquemas ER que se explicó en el capítulo 4. Subraya la obligación del cumplimiento de las normas para asignar nombres y la resolución de sinónimos y homónimos al principio del proceso de diseño.

Diseño lógico en el modelo de redes

En este capítulo y en el próximo se tratarán los otros dos modelos de datos que son importantes comercialmente además del modelo relacional: el modelo de redes y el jerárquico. Debido a su aceptación en la industria, se han diseñado y comercializado muchos productos de sistemas de gestión de bases de datos que siguen estos modelos. Los sistemas comerciales actuales que usan el modelo de redes incluyen IDS II (Integrated Data Store, almacén de datos integrado) de Honeywell, DBMS-11 y VAX-DBMS de Digital, IDMS de Cullinet (ahora Computer Associates), DMS 1100 de Univac e IMAGE de Hewlett-Packard, por nombrar los más importantes. Este capítulo trata el modelo de redes *sin* referirse a detalles específicos de los DBMS individuales; en vez de ello, se analiza el modelo y las correspondencias de manera general.

El apartado 13.1 repasa los conceptos y recursos básicos del modelo de redes. El apartado 13.2 analiza la correspondencia de los esquemas ER con el modelo de redes; se consideran las correspondencias de los distintos componentes de modelado en forma independiente, y luego se ve un ejemplo. El apartado 13.3 trata la correspondencia de las consultas o solicitudes de actualización en la especificación de los accesos de navegación de alto nivel, con los recursos de manipulación de datos del modelo de redes. El apartado 13.4 vuelve a la base de datos del caso de estudio y hace corresponder su diseño conceptual ER con un diseño lógico basado en el modelo de redes. También se muestran las correspondencias de algunos ejemplos de esquemas de navegación con el lenguaje de manipulación de datos del modelo de redes para ilustrar las operaciones de recuperar, insertar y eliminar de la base de datos. El apartado 13.5 trata la retroingeniería desde el esquema del modelo de redes a un esquema ER. Esta abstracción es valiosa para entender los datos que ya existen en una base de datos de red.

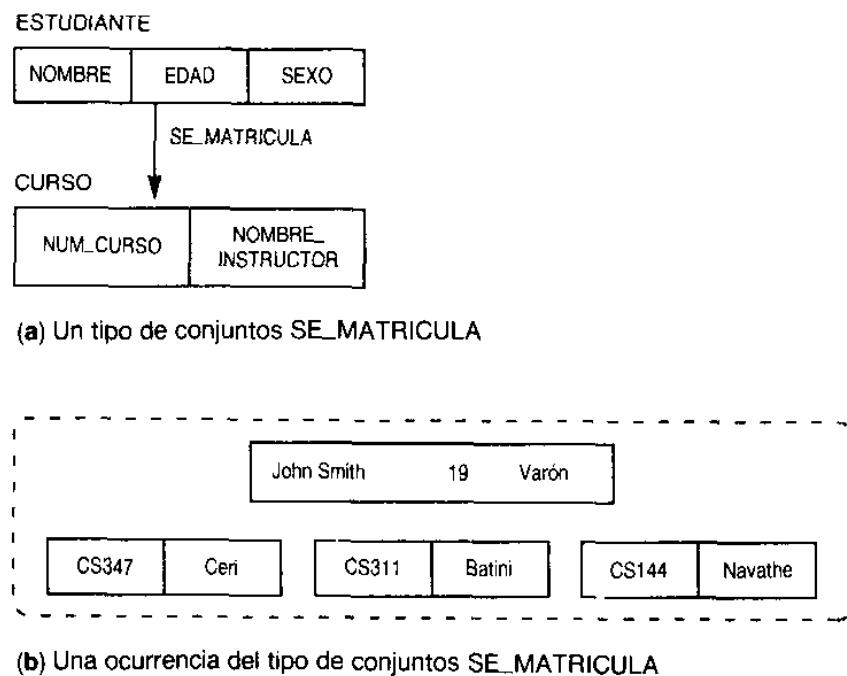


Figura 13.1. Un tipo de conjuntos y su ocurrencia.

13.1. El modelo de redes

El modelo de datos de redes, también conocido como modelo DBTG o CODASYL, se basa en el informe CODASYL DBTG de 1971. Este informe presentó recomendaciones para un modelo de datos y un sistema de bases de datos que desde entonces han sido modificados (en 1978 y 1981). Los diversos DBMS antes mencionados han puesto en práctica las distintas versiones del modelo CODASYL DBTG de acuerdo con estos tres informes. Aquí describiremos los conceptos generales del modelo de redes sin hacer referencia a ningún informe individual.

Hay dos conceptos básicos de estructuras en el modelo de redes: tipos de registros y tipos de conjuntos. Cada **tipo de registros** describe la estructura de un grupo de registros que almacenan el mismo tipo de información. Un **tipo de conjuntos** es una interrelación de uno a muchos entre dos tipos de registros. Los casos de un tipo de registros se denominan **registros**; cada registro representa alguna información del mundo real acerca de un grupo de elementos, denominados **elementos de datos** o **atributos** de ese registro. Para un tipo de conjuntos dado, **una ocurrencia del conjunto** (o **caso del conjunto**) es una colección de registros que contiene un registro del tipo de registros *propietario*, y muchos registros del tipo de registros *miembro*.

La figura 13.1 muestra dos tipos de registros ESTUDIANTE y CURSO, con el tipo de conjuntos SE_MATRICULA entre ellos, con ESTUDIANTE como el tipo de registros propietario y CURSO como el tipo de registros miembro. La representación

CALIFICACIONES_EXAMENES

NOMBRE_ESTUD.	NUM_CURSO	(CALIFIC.)
---------------	-----------	------------

(a) Tipo de registros CALIFICACIONES_EXAMENES, con el elemento de datos vector

CONDUCTOR

NSS	NUM_LIC_CONDUCE	(COCHES)			
		N_MATR	MARCA	AÑO	COLOR

(b) Tipo de registros CONDUCTOR, con grupo de repetición COCHES

Figura 13.2. Uso de vectores y grupos de repetición en el modelo de redes.

diagramática mostrada en la que una flecha va del registro propietario al registro miembro se denomina **diagrama Bachman**, por Charles Bachman, que fue el primero en introducirla. La ocurrencia del conjunto incluye el registro propietario de John Smith y tres registros miembros que corresponden a los tres cursos en los cuales está matriculado. Una base de datos puede contener muchas ocurrencias del tipo de conjuntos SE_MATRICULA, una por estudiante. Obsérvese que si un estudiante no se matricula en ningún curso, la ocurrencia del registro ESTUDIANTE de todos modos una ocurrencia del conjunto en la que hay un registro propietario y cero registros miembros. La figura muestra los elementos de datos contenidos en los registros antes mencionados.

Los sistemas del modelo de redes permiten **vectores**, que son elementos de datos que pueden tener valores múltiples dentro de un registro. Esto corresponde a un atributo polivalente en el modelo ER. De manera similar, los **grupos de repetición** permiten la inclusión de un grupo de valores (para un grupo de atributos distintos) y también permiten que ese grupo se repita (tenga múltiples ocurrencias) dentro de un registro. Esta situación corresponde a los atributos compuestos polivalentes en la terminología del modelo ER o del relacional. La figura 13.2 muestra el tipo de registros CALIFICACIONES_EXAMENES que contiene un vector CALIFICACIONES, el cual puede tener múltiples valores de calificaciones. Para distinguir un elemento de datos con valores múltiples, se encierra entre paréntesis. El tipo de registros CONDUCTOR incluye un grupo de repetición denominado COCHES, que es un tipo compuesto de los elementos de datos NUM_MATRICULA, MARCA, AÑO y COLOR. Puede haber varios coches dentro de un caso del registro CONDUCTOR, cada uno con valores para los cuatro elementos de datos.

La noción de conjunto (o caso de conjunto) en el modelo de redes difiere de la noción matemática de conjunto en dos aspectos importantes: 1) el caso de conjunto tiene un **elemento distinguido** (el registro propietario), mientras que en un conjunto matemático no hay distinción entre los miembros de un conjunto;

2) los registros miembros dentro de una ocurrencia de conjunto están *ordenados* en el modelo de redes, mientras que el orden es indiferente en un conjunto matemático. Por estas razones a veces se llama al conjunto del modelo de redes conjunto **acoplado al propietario o co-conjunto**.

13.1.1. Propiedades de los tipos de conjuntos

Una definición de tipo de conjuntos incluye las siguientes especificaciones:

1. El nombre del tipo de conjuntos.
2. El nombre del tipo de registros propietario.
3. El nombre del tipo de registros miembro.
4. La opción de *orden del conjunto*, que especifica cómo deben ordenarse los registros miembros *dentro de una ocurrencia de conjunto*. Las posibles opciones son:
 - a) Clasificados por un campo de ordenamiento.
 - b) Por defecto: el sistema decide arbitrariamente.
 - c) Primero o último: el miembro recién insertado se coloca primero o último en relación a los miembros que ya están en la lista.
 - d) Siguiente o anterior: el miembro recién insertado se coloca inmediatamente antes o después del miembro *actual* del conjunto, es decir, el que la aplicación procesó más recientemente.
5. Opción de *especificación de selección del conjunto*, que especifica cómo se debe seleccionar una ocurrencia de conjunto. Las posibles opciones son:
 - a) Estructural: aquí la selección de un propietario al cual un registro miembro debe estar vinculado se determina tomando el valor de un campo específico del registro miembro y comparándolo con un valor del campo correspondiente en el registro propietario.
 - b) Por aplicación: aquí la decisión para elegir una ocurrencia de conjunto se deja a la aplicación, que es responsable de hacer que la ocurrencia de conjunto sea *actual* (el concepto de **actualidad**, refiriéndose a la actualidad de los tipos de registros y de conjuntos, se explica posteriormente en el apartado 13.3), de modo que cualquier acción se aplique automáticamente a esa ocurrencia de conjunto.
 - c) Por valor de <nombre de campo> IN <nombre de tipo de registros> : en esta opción, el nombre de un campo se suministra junto con el nombre del registro propietario; así el sistema puede localizar la ocurrencia de conjunto apropiada localizando primero un registro propietario que coincida con el valor dado.
6. Opción de *inserción de conjunto*. Cuando se inserta un registro utilizando el mandato STORE, esta opción decide qué sucede con los tipos de conjuntos de los cuales este registro es miembro. Hay dos opciones:
 - a) Automática: El registro miembro nuevo se inserta automáticamente en

una ocurrencia de conjunto apropiada, la cual se determina parcialmente por la especificación de selección de conjunto.

- b) Manual: El programador debe conectar manualmente el registro con aquellos tipos de conjuntos de los cuales se ha declarado miembro. Sin embargo, la decisión sobre si añadir o no un registro a un tipo de conjuntos en particular y la determinación de un registro propietario apropiado se deja al programador de la aplicación.

7. Opción de *eliminación de conjunto (retención)*. Cuando hay que eliminar un registro de una ocurrencia de conjunto con los mandatos ERASE o DELETE, esta opción decide: 1) si tal eliminación está permitida, o 2) si el registro debe quedarse permanentemente en ese tipo de conjuntos, o 3) si el registro debe conectarse con algún otro conjunto. Las opciones son:

- a) Opcional: El registro miembro puede existir por sí mismo sin ser miembro de ninguna ocurrencia del conjunto; se puede conectar (con CONNECT) o desconectar (con DISCONNECT) libremente de ocurrencias de conjuntos.
- b) Obligatorio: El registro miembro no puede existir por sí mismo; debe estar conectado a alguna ocurrencia del tipo de conjuntos. La eliminación de una ocurrencia de conjunto y conexión posterior a otra ocurrencia puede realizarse mediante una simple operación RECONNECT¹.
- c) Fijo: Como en la opción obligatoria, el registro miembro no puede existir por sí mismo; una vez que se conecta a alguna ocurrencia del conjunto, queda fijo; no puede reconectarse a otra ocurrencia de conjunto. La única manera de eliminarlo es suprimiendo su propietario, de manera que él sea eliminado automáticamente.

8. Opciones de *modalidad* o de *implantación*. Los diferentes sistemas proveen diferentes opciones para la implantación de conjuntos: Las opciones típicas son:

- a) Lista circular o representación en anillo: los registros dentro de una ocurrencia de conjunto se colocan en una lista circular. La lista puede estar doblemente enlazada para permitir un recorrido hacia adelante o hacia atrás.
- b) Arrays de punteros: el registro propietario contiene un array de punteros para todos los registros miembros.
- c) Conjuntos indizados: para cada ocurrencia de conjunto, se crea un índice y se almacena con el registro propietario. Usando este índice, los miembros pueden ser localizados desde el propietario conociendo un valor del campo de indización.

9. Opciones adicionales de punteros: Para acelerar los recorridos entre miembros y propietarios de los conjuntos, se puede especificar punteros adicionales (un puntero de propietario que va desde un miembro a un propietario,

¹ CONNECT, DISCONNECT y RECONNECT son mandatos del lenguaje de tratamiento de datos de redes. Se usan para realizar las acciones correspondientes sobre un registro miembro y con relación a un conjunto.

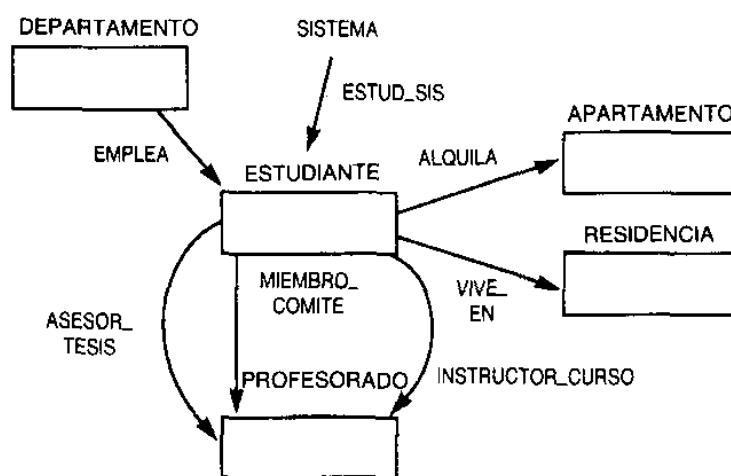


Figura 13.3. Empleo de tipos de conjuntos múltiples que utilizan el mismo tipo de registros.

primeros o últimos punteros del propietario al primer o último miembro, etc.).

13.1.2. Restricciones de los conjuntos

Dos restricciones principales limitan la libertad de modelado usando los tipos de registros y los tipos de conjuntos. Primero, un registro dado puede ser miembro de una sola ocurrencia de un tipo de conjuntos dado. Esto hace que todo conjunto muestre una estricta interrelación de uno a muchos entre los tipos de registros propietarios y los tipos de registros miembros. Segundo, un conjunto no puede tener el mismo tipo de registros como propietario y como miembros. Un conjunto así se denominaría **conjunto recursivo**, y está prohibido. Por otra parte, en el modelo de redes existen los siguientes rasgos:

1. Un tipo de registros dado puede ser propietario en varios tipos de conjuntos diferentes.
2. Un tipo de registros dado puede ser miembro en varios tipos de conjuntos diferentes.
3. En virtud de lo anterior, es posible definir varios tipos de conjuntos entre el mismo par de tipos de registros que representen relaciones diferentes entre ellos.

La figura 13.3 muestra el tipo de registros **ESTUDIANTE** que participa en varios tipos de conjuntos. Obsérvese que los tipos de conjuntos **ASESOR_TESIS**, **MIEMBRO_COMITE** e **INSTRUCTOR_CURSO** relacionan a **ESTUDIANTE** con **PROFESORADO**. Además, un estudiante puede alquilar un apartamento o vivir en una residencia pero no las dos cosas. Esta última restricción de exclusión no puede

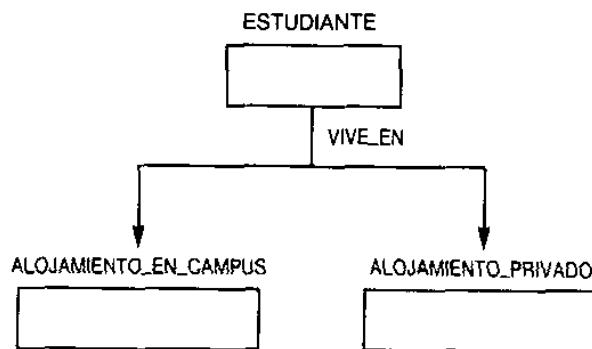


Figura 13.4. Un conjunto multimiembro.

ser impuesta automáticamente por el modelo; imponerla es responsabilidad de los programas de aplicación.

13.1.3. Tipos de conjuntos especiales

Vale la pena mencionar dos tipos de conjuntos especiales.

Conjuntos propiedad del sistema o conjuntos singulares. Este tipo de conjuntos no tiene tipo de registros propietario y se define solamente con el propósito de proveer un *ordenamiento de todos los casos* de un tipo de registros dado. El conjunto ESTUD-SIS de la figura 13.3 puede usarse para tener acceso a los estudiantes en orden por número de seguro social, si está ordenado según ese número. Se considera al sistema como propietario ficticio de dicho conjunto; de ahí el nombre, *conjunto propiedad del sistema*. También se usan para proporcionar **puntos de entrada** a la base de datos.

Conjuntos multimiembros. Estos conjuntos se usan en casos donde más de un tipo de registros miembro participa en una interrelación. La figura 13.4 muestra un conjunto multimiembro denominado VIVE_EN, donde el registro miembro puede ser ALOJAMIENTO_EN_CAMPUS o bien ALOJAMIENTO_PRIVADO. Obsérvese que esto es preferible a crear dos tipos de conjuntos diferentes (VIVE_EN y ALQUILA), como se hizo en la figura 13.3 para representar información similar. Este tipo de recurso se suministra en muy pocos DBMS de red.

13.2. Correspondencia de esquemas de modelo ER al modelo de redes

En este apartado se presenta una metodología para el diseño lógico usando el modelo de redes como modelo objetivo. Se supone una vez más que el esquema

inicial es un esquema ER como el producido en la figura 11.5. El esquema resultante tiene la forma de un conjunto de tipos de registros y tipos de conjuntos. No se considerarán las especificaciones detalladas para la inserción y retención de conjuntos. Los tipos de transformaciones considerados son los mismos que se trataron en el capítulo 12 para el modelo relacional.

13.2.1. Correspondencia de atributos compuestos y polivalentes

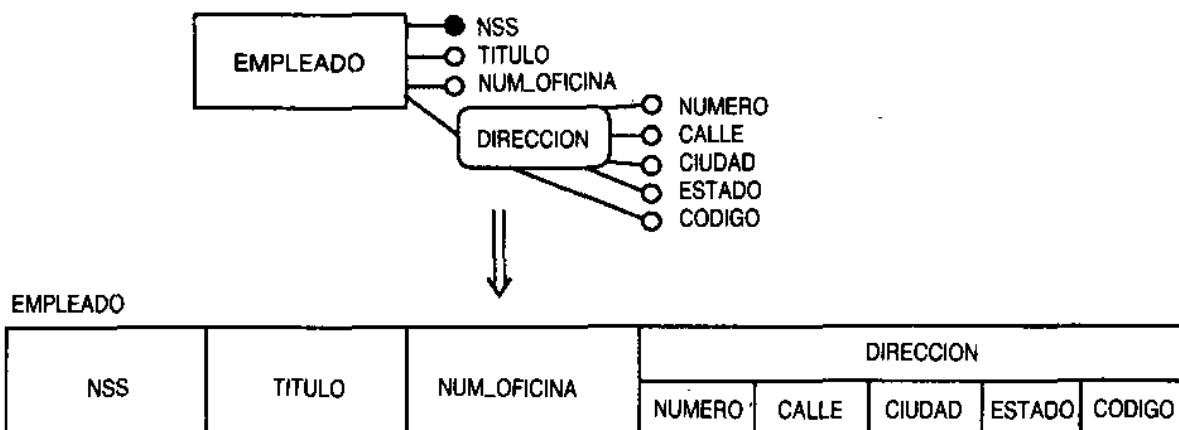
El modelo de redes tiene la ventaja de utilizar atributos compuestos y polivalentes directamente como parte del modelo. Un atributo compuesto como DIRECCION puede representarse como elemento de datos y recibir un nombre. Este elemento puede ser desglosado posteriormente en sus elementos de datos componentes, esto es, NUMERO, CALLE, CIUDAD, ESTADO y CODIGO_POSTAL. Tal grupo de elementos de datos se conoce normalmente como **grupo de repetición** en los sistemas de redes². La figura 13.5a muestra gráficamente el grupo de repetición DIRECCION como una parte del registro EMPLEADO. Obsérvese que dicho grupo, aunque se califique como grupo de repetición, no tiene necesariamente que repetirse.

La mayoría de las implantaciones del modelo de redes requieren una especificación del *factor de repetición* de cada grupo de repetición para designar el número máximo de veces que se puede repetir dentro de una ocurrencia de un registro. En un caso extremo este número puede ser 1. Por ejemplo, dentro de un registro denominado EMPLEADO, el grupo de repetición DIRECCION podría ocurrir sólo una vez porque se puede desear mantener una sola dirección, la de la residencia principal, para cada EMPLEADO. En los sistemas orientados a registros de longitud fija, el factor de repetición puede causar desperdicio de almacenamiento si la información de repetición varía considerablemente entre las ocurrencias del registro que las contiene. Por ejemplo, si los dependientes de un empleado se modelan como grupo de repetición, es difícil asignarle un factor de repetición máximo. En tales casos es mejor modelar el grupo como una entidad aparte.

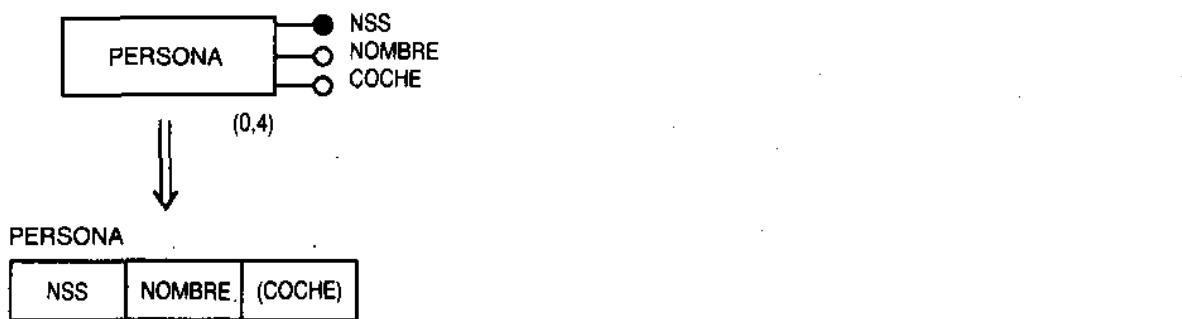
Siempre que un solo atributo o grupo de atributos pueda recibir valores múltiples, será posible usar los recursos de vectores y grupos de repetición en los sistemas de redes. Por ejemplo, la entidad PERSONA puede tener un atributo COCHE que contiene el número de matrícula para el coche que posee esa persona. Si se desea permitir uno o más coches por persona, el atributo COCHE se convierte en polivalente en el modelo ER; lo que se representa como un elemento de datos de vector en el modelo de redes. El sistema puede permitir una especificación del número máximo de valores para un elemento de datos de vector (por ejemplo, cuatro para el atributo COCHE en el registro PERSONA de la figura 13.5b).

Hasta ahora hemos visto un elemento *compuesto* DIRECCION representado

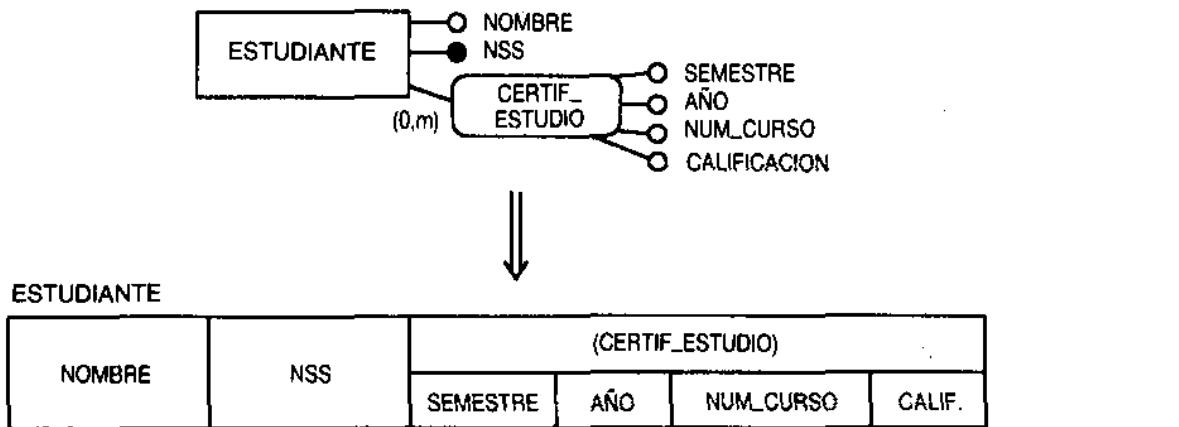
² Se usa el término *sistema de redes* para referirse a cualquier implantación del modelo de datos de redes. Por lo general se refiere a un DBMS que sigue la versión CODASYL del modelo de redes.



(a) Atributo compuesto DIRECCION representado por un grupo de repetición



(b) Atributo polivalente COCHE representado por un vector



(c) Atributo compuesto y polivalente CERTIF_ESTUDIO representado como grupo de repetición

Figura 13.5. Transformación de atributos compuestos y polivalentes en el modelo de redes.

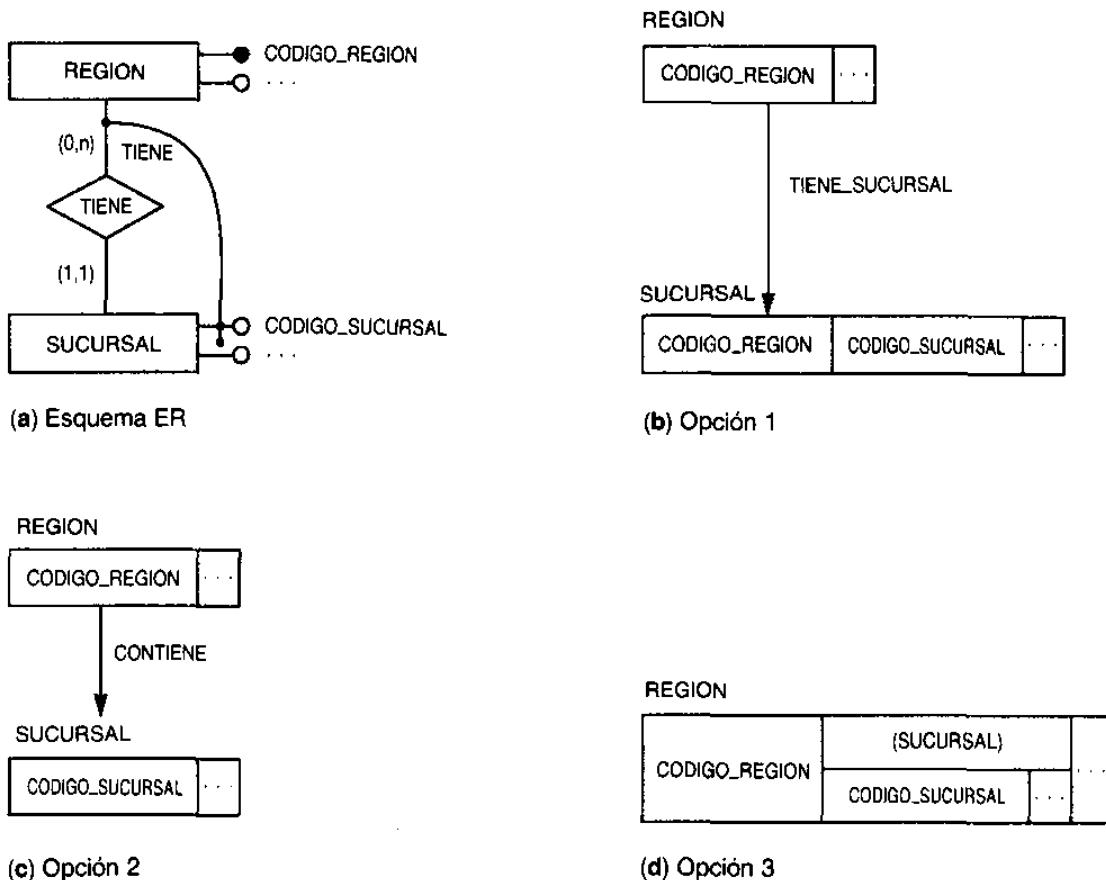


Figura 13.6. Modelado de identificadores externos en el modelo de redes.

como un grupo de repetición y un elemento *polivalente* COCHE representado como un vector. Algunas veces se necesita representar elementos que sean compuestos y polivalentes. La figura 13.5c muestra un atributo compuesto polivalente denominado CERTIF_ESTUDIO que se modela como un grupo de repetición. Para un ESTUDIANTE dado, hay muchas ocurrencias de CERTIF_ESTUDIO dentro del mismo registro. Otra manera de utilizar CERTIF_ESTUDIO es crear un tipo de registros aparte y definir un tipo de conjuntos con ESTUDIANTE como propietario y CERTIF_ESTUDIO como miembro.

13.2.2. Correspondencia de identificadores externos

Los identificadores externos pueden tratarse de diferentes maneras en el modelo de redes. Considérese el ejemplo de la figura 13.6, donde se muestran solamente atributos importantes. Aquí, el identificador externo CODIGO_REGION de la entidad REGION identifica la entidad SUCURSAL además de su propio identificador CODIGO_SUCURSAL, como se muestra en la figura 13.6a. Para resolver esto hay

tres opciones en el modelo de redes. Primero, se puede crear dos tipos de registros, REGION y SUCURSAL, y efectivamente incluir el identificador CODIGO_REGION en el tipo de registros SUCURSAL. Definamos el conjunto TIENE_SUCURSAL de REGION a SUCURSAL. Ahora se puede definir una restricción en este conjunto especificada por:

```
SET SELECTION IS STRUCTURAL CODIGO_REGION IN REGION =
  CODIGO_REGION IN SUCURSAL
```

para designar que CODIGO_REGION debe coincidir en los dos registros dentro de la misma ocurrencia de conjunto. Con base en esta restricción estructural, es posible seleccionar la opción de inserción automática para este conjunto.

Segundo, se puede crear los dos registros pero *no incluir* el elemento CODIGO_REGION en el registro miembro SUCURSAL. En tal caso se insertaría manualmente una SUCURSAL en el conjunto anterior; es responsabilidad del programador de la aplicación garantizar que una sucursal esté insertada en la región apropiada. Aquí la identificación externa se aplica *implícitamente* en virtud de la pertenencia al propietario correcto en la ocurrencia de conjunto correcta.

Una tercera posibilidad es hacer a SUCURSAL un grupo de repetición dentro de REGION, como en la figura 13.6d. Esto es deseable sólo si se tienen pocas sucursales en un distrito y no hay demasiados datos relacionados con cada SUCURSAL. Si SUCURSAL tiene una gran cantidad de datos propios y el sistema exige siempre que se especifique un número máximo, fijo, para las ocurrencias del grupo de repetición, se puede desperdiciar mucho espacio para las sucursales no asignadas a una región. Por ello, en los sistemas de redes se adopta esta solución sólo si la información de sucursal se considera una parte integral de la información regional. Una ventaja de esta solución es que no se necesita obtener acceso a un conjunto adicional para obtener la información de sucursal.

13.2.3. Correspondencia de entidades

El proceso de correspondencia de entidades es similar en los modelos de redes y relacionales. Cada entidad se transforma en un tipo de registros, y los atributos de la entidad se convierten en los elementos de datos del registro. La figura 13.7 muestra un esquema ER para una aplicación de procesamiento de pedidos, y la figura 13.8 muestra su correspondencia con un esquema de redes. Las entidades (REGION, SUCURSAL, PRODUCTO, etc.) se transforman en los tipos de registros correspondientes (con los mismos nombres por conveniencia) en la figura 13.8

13.2.4. Transformación de interrelaciones de uno a uno

Se considerarán las correspondencias de las interrelaciones del modelo ER en el modelo de redes tratando primero las interrelaciones binarias. Entre éstas, hay

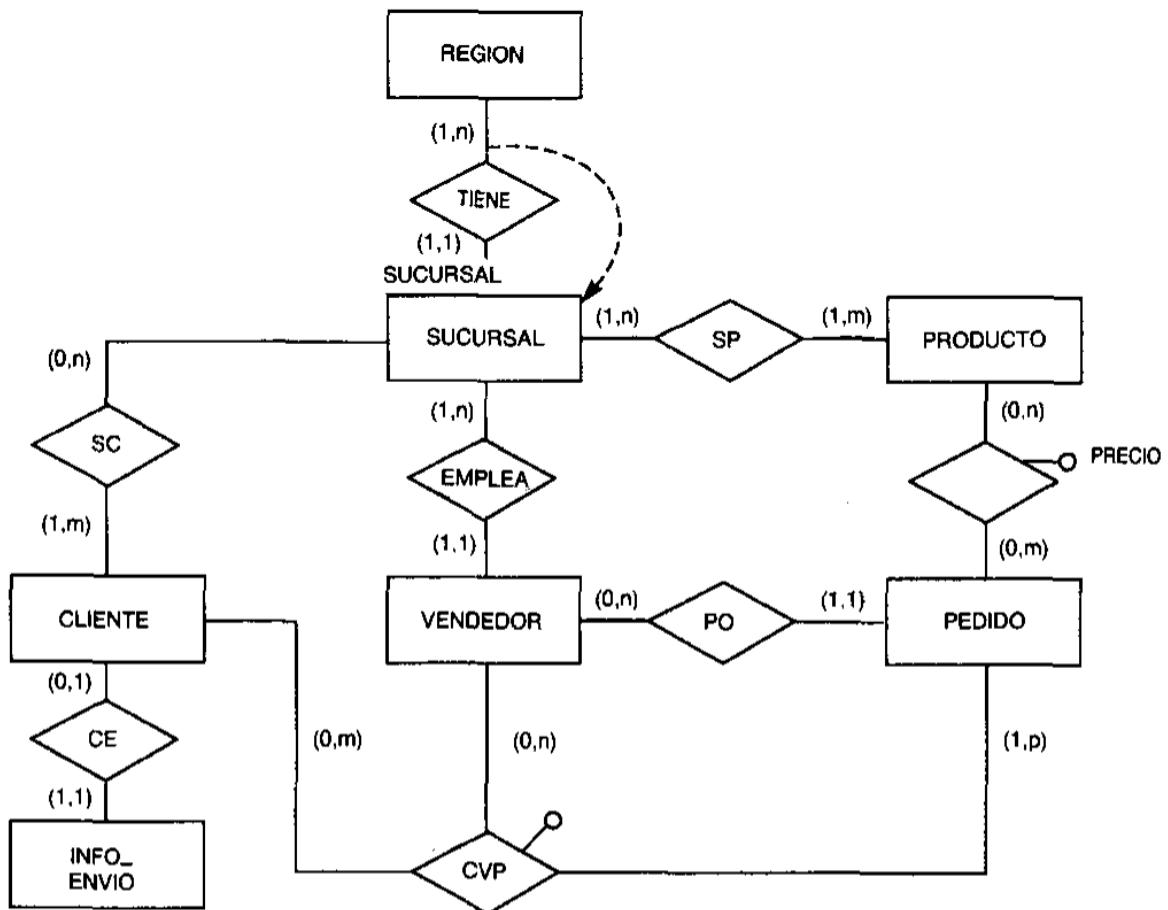


Figura 13.7. Base de datos de control de pedidos en el modelo ER.

tres razones de cardinalidad por considerar: uno a uno, uno a muchos y muchos a muchos. Una interrelación uno a uno se muestra en la figura 13.7 entre CLIENTE e INFO_ENVIO. Por lo general, las interrelaciones uno a uno no deben tener atributos, ya que pueden ser considerados como parte de cualquiera de las dos entidades. Hay que considerar dos opciones.

Primero, se puede integrar los dos registros correspondientes en un tipo de registros. Esta opción tiene sentido si los dos tipos de registros participan en el mismo conjunto de otras interrelaciones o en ninguna otra interrelación. Además, se supone que la participación de las dos entidades en la interrelación es obligatoria. Hay dos posibilidades:

1. Las dos entidades tienen las mismas claves primarias. Supongamos que tanto CLIENTE como INFO_ENVIO tienen la clave primaria NUM_CLIENTE. En este caso los dos registros correspondientes se integran en un registro combinando todos los atributos e incluyendo la clave primaria sólo una vez.
2. Las entidades tienen claves primarias distintas. Supóngase que CLIENTE e IN-

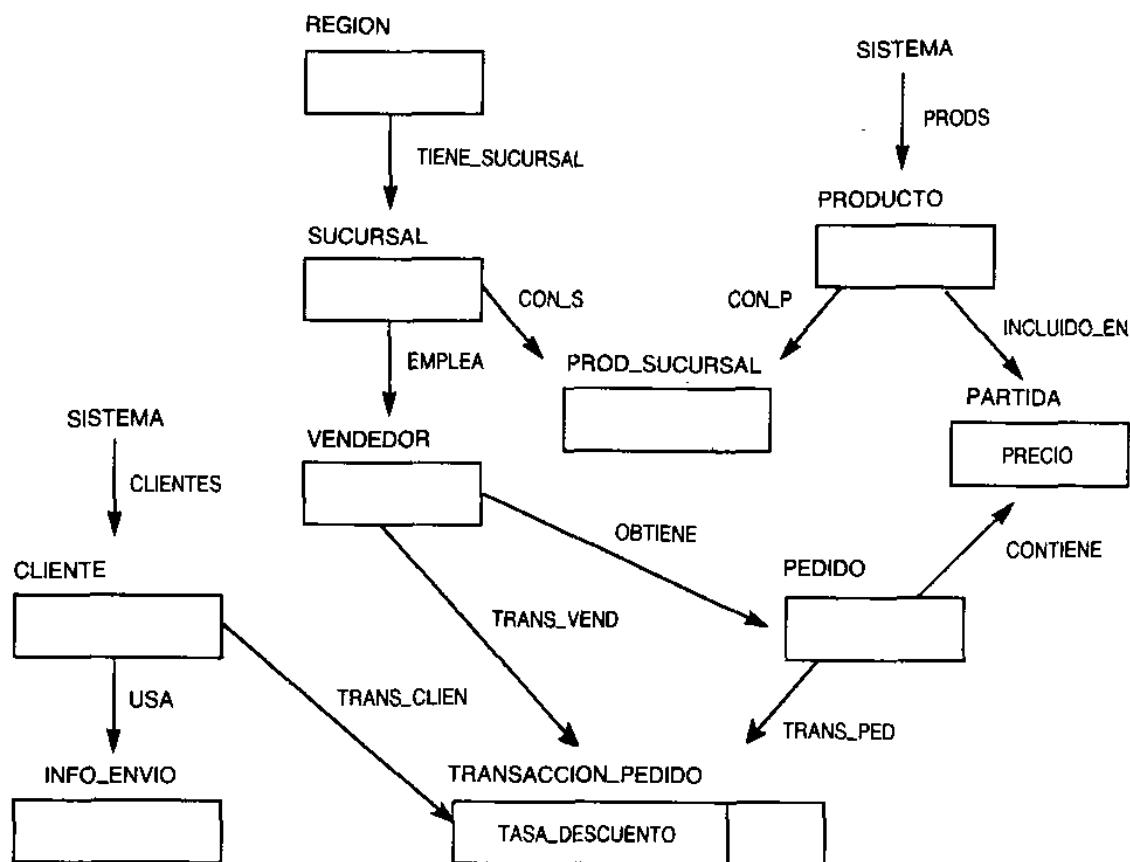


Figura 13.8. Esquema de la base de datos de control de pedidos en el modelo de redes.

FO_ENVIO tienen claves primarias distintas, digamos NUM_CLIENTE y CODIGO POSTAL, CALLE, NUM_CASA, respectivamente. En este caso también se integran en un registro combinando todos los atributos e incluyendo las dos claves primarias. Una de las dos claves primarias será conservada como la clave del registro resultante; en el presente ejemplo, sería NUM_CLIENTE.

La segunda opción es definir un tipo de conjuntos. Esta opción es particularmente útil si los dos tipos de registros implicados participan en otras interrelaciones diferentes o si la participación de al menos una entidad en la interrelación es opcional. En este caso se define un tipo de conjuntos entre los dos tipos de registros. Si las dos entidades tienen una participación obligatoria en la interrelación, se puede declarar a cualquiera de ellas como miembro del conjunto. Si una entidad tiene participación obligatoria y la otra no, la primera debe ser escogida como registro miembro. Por ejemplo, si no todo CLIENTE tiene INFO_ENVIO, pero cada caso de INFO_ENVIO debe tener un CLIENTE asociado con ella, lo apropiado es definir el conjunto con INFO_ENVIO como miembro. Así es como se define el conjunto USA en la figura 13.8. Obsérvese que cuando se de-

fine un conjunto así, es responsabilidad de la aplicación imponer la cardinalidad máxima de 1 en el caso anterior; esto es, que no debe permitirse añadir más de un miembro al conjunto USA; sin embargo, el modelo de redes no tiene un mecanismo inherente para resolver la restricción de cardinalidad máxima.

13.2.5. Transformación de interrelaciones de uno a muchos

En la figura 13.7 se muestra una interrelación de uno a muchos entre VENDEDOR y PEDIDO. Por lo general, una interrelación de uno a muchos no debe tener ningún atributo; si se ha usado atributos, primero deben transferirse a la entidad en el lado de «muchos» de la interrelación y captarse en el tipo de registros correspondiente. La interrelación entonces se transforma en un tipo de conjuntos, con la entidad en el lado de «uno» como propietario. La interrelación anterior VP se transforma en el conjunto OBTIENE de la figura 13.8.

13.2.6. Transformación de interrelaciones de muchos a muchos

En la figura 13.7 se muestra una interrelación de muchos a muchos entre PRODUCTO y PEDIDO. Una interrelación de muchos a muchos representa el caso más general y, por consiguiente, puede tener atributos. Sea que se haya usado atributos o no, el método más común es transformar la interrelación primero en un tipo de registros; este registro de interrelación se denomina **registro vínculo** o enlace. Cualquier atributo de la interrelación se convierte en elemento de datos de este registro. Entonces se definen *dos tipos de conjuntos*, cada uno de los cuales contiene este registro vínculo como registro miembro; los tipos de registros correspondientes a las entidades originales se convierten en propietarios de esos conjuntos. La figura 13.8 ilustra la situación con el registro PARTIDA que se crea como un registro vínculo para representar la interrelación PP. Los dos tipos de conjuntos CONTIENE e INCLUIDO_EN se definen a partir de PEDIDO y PRODUCTO, respectivamente, con partida como miembro. El efecto neto es emplear una interrelación de muchos a muchos por medio de dos tipos de conjuntos de uno a muchos. Los atributos de la interrelación, como es PRECIO, pertenecen legítimamente al registro PARTIDA.

Es posible otra opción cuando se está seguro de que la interrelación no tiene atributos. Consiste en definir dos tipos de conjuntos, entre los dos registros en cuestión, en direcciones opuestas; esto es, las funciones de propietario/miembro se invierten en los dos conjuntos. En el ejemplo anterior, esto lleva a definir un conjunto con PEDIDO como propietario, de manera que cada PRODUCTO pueda ser relacionado con un PEDIDO a través de este conjunto. El otro conjunto relaciona todos los PEDIDO a un PRODUCTO dado como miembros. Naturalmente, atributos como PRECIO *no pueden* representarse. Esta opción no se usa normalmente porque impide la adición futura de atributos a la interrelación. Además, el uso de dos tipos de conjuntos para representar la misma interrelación entre

un par de tipos de registros introduce una redundancia innecesaria y un gasto adicional para asegurar la congruencia o consistencia.

13.2.7. Observaciones generales sobre la transformación de interrelaciones binarias

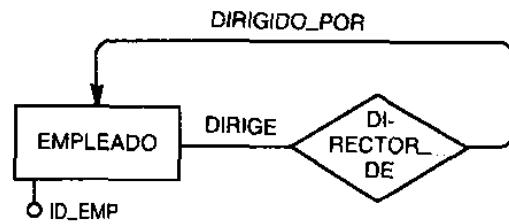
Los apartados precedentes describen la transformación de interrelaciones binarias con diferentes razones de cardinalidad. Hay unas pocas observaciones que se aplican a estas transformaciones en general. Primero, se puede duplicar arbitrariamente uno o más atributos de un tipo de registros propietario de un tipo de conjuntos –sea que represente una interrelación uno a uno o de muchos a muchos– en el tipo de registros miembro. Esto en general lo determinan los requisitos de la aplicación, de manera que se pueda evitar un acceso adicional (una operación de lectura, GET) al registro propietario. Segundo, si los atributos duplicados corresponden a la clave primaria del propietario, se le puede imponer al conjunto una restricción estructural: SET SELECTION IS STRUCTURAL... Esta restricción sirve para localizar automáticamente la ocurrencia apropiada del conjunto, dada la ocurrencia del miembro. Finalmente, recuérdese que las restricciones de cardinalidad mínima y máxima aplicables a las interrelaciones en el esquema ER deben ser impuestas por las aplicaciones explícitamente.

13.2.8. Transformación de interrelaciones n-arias.

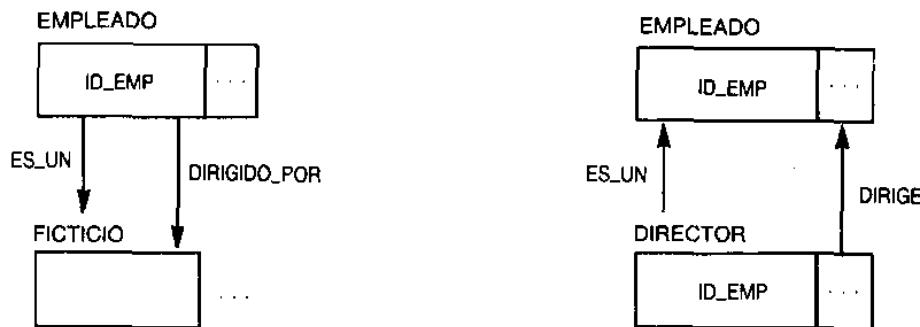
Consideremos las interrelaciones n-arias, con $n > 2$. Aquí la transformación es similar a la de las interrelaciones binarias de muchos a muchos empleando la primera opción. Primero se transforma la interrelación en un tipo de registros para crear un registro de interrelación denominado **registro vínculo**. Cualesquier atributos de la interrelación n-aria se convierten en elementos de datos dentro de este registro. En seguida se definen *n tipos de conjuntos*, cada uno de los cuales contiene este registro vínculo como registro miembro; los tipos de registros que corresponden a las entidades originales se convierten en propietarios de estos conjuntos. La figura 13.7 ilustra la interrelación n-aria CVP entre las tres entidades CLIENTE, VENDEDOR y PEDIDO. Esta se traduce en el registro vínculo correspondiente, TRANSACCION_PEDIDO en la figura 13.8. El atributo de la interrelación, denominado TASA_DESCUENTO, se coloca dentro de este registro. Se definen tres tipos de conjuntos con los tres tipos de registros anteriores como propietarios, y todos con el registro vínculo como miembro. El efecto neto es utilizar una interrelación n-aria por medio de *n tipos de conjuntos* de uno a muchos.

13.2.9. Transformación de interrelaciones recursivas

Considérese la interrelación recursiva o de anillo denominada DIRECTOR_DE entre una entidad EMPLEADO y ella misma. Es una interrelación de uno a mu-



(a) Una interrelación recursiva en el esquema ER



(b) Dos maneras de tratar la interrelación recursiva

Figura 13.9. Transformación de una interrelación recursiva.

chos, en cuanto a que un empleado es director de muchos empleados. Esta interrelación se emplea creando un registro vínculo *ficticio* y definiendo dos tipos de conjuntos para relacionar este registro ficticio al registro original. Los dos tipos de conjuntos designan los dos papeles que desempeñan las entidades denominadas DIRIGE y DIRIGIDO_POR que se muestran en la figura 13.9a.

Como ilustración, la figura 13.9 muestra dos alternativas. La primera muestra al director como registro vínculo FICTICIO sin ningún atributo, el conjunto ES_UN es 1:1 y está presente solamente cuando un empleado es director; el conjunto DIRIGIDO_POR es 1:n y está presente para todos los empleados que son dirigidos por otro empleado. En la segunda alternativa, un registro ficticio denominado DIRECTOR aparece con la clave ID_EMP para hacerlo accesible independientemente. Los dos conjuntos ES_UN y DIRIGE se muestran con DIRECTOR como propietario. Obsérvese que también son posibles otras permutaciones similares a los conjuntos ES_UN y DIRIGIDO_POR.

13.2.10. Adición de conjuntos propiedad del sistema

Finalmente, se pueden añadir algunos conjuntos propiedad del sistema al esquema final de redes con el propósito de obtener acceso a registros de algunos tipos. En el ejemplo de la figura 13.8 se muestran los conjuntos propiedad del

Tabla 13.1. Resumen de los mandatos del DML de redes

Mandato	Función
<i>Recuperación</i>	
GET	Se usa para recuperar el registro actual y colocarlo en la variable correspondiente de área de trabajo del usuario
<i>Navegación</i>	
FIND	Se usa para localizar un registro y establecer los indicadores de actualidad del tipo de registros implicado y de los conjuntos relacionados
<i>Actualización de registros</i>	
STORE	Almacena el registro nuevo en la base de datos y lo hace el registro actual
ERASE	Borra de la base de datos el caso «actual» del tipo de registros
MODIFY	Modifica algunos campos del caso actual del tipo de registros
<i>Actualización de conjuntos</i>	
CONNECT	Conecta un registro miembro a un caso de conjunto
DISCONNECT	Elimina un registro miembro de un caso de conjunto
RECONNECT	Mueve un registro miembro de un caso de conjunto a otro

sistema CLIENTES (para el tipo de registros CLIENTE) y PRODS (para el tipo de registros PRODUCTO).

13.3. Correspondencia de operaciones del modelo ER al modelo de redes

En este apartado se muestran algunos ejemplos de transformación de las especificaciones de esquemas de navegación del modelo ER a las construcciones típicas del lenguaje de manipulación de datos de redes (DML). No se intenta describir los DML de redes en detalle. Los lectores interesados pueden consultar Elmasri y Navathe (1989, Cap. 11; véase la bibliografía). Un estándar denominado dnl fue propuesto por ANSI en 1986, pero no tiene muchos seguidores hasta ahora. Los mandatos del DML de redes se resumen en la tabla 13.1.

El DML de redes es un lenguaje orientado a la navegación. Una recuperación implica moverse de registro en registro (usando mandatos FIND) entre los conjuntos hasta localizar el registro que se requiere. Sólo se puede leer un registro (usando una operación GET) después de haberlo localizado. Para la actualización de los registros se utiliza una estrategia similar de navegar a través de los

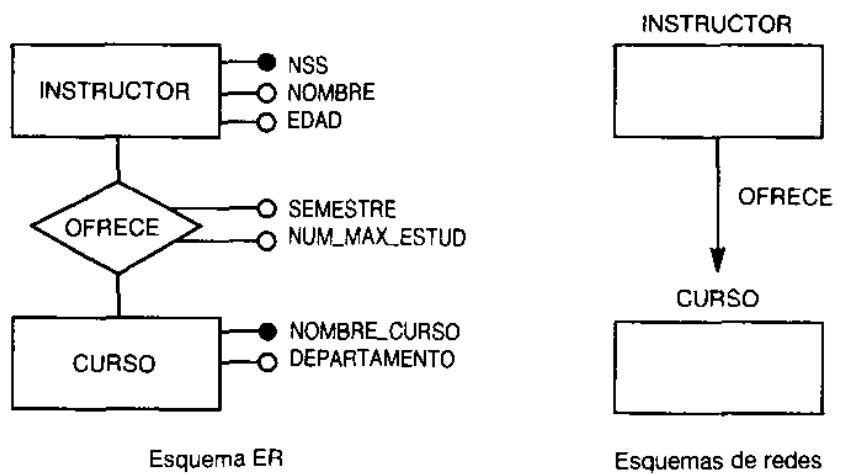
tipos de registros y de conjuntos hasta encontrar los registros por actualizar. El control real de la navegación, la verificación de los códigos de estado, etc., es tarea del programa de aplicación. Se usa DB_STATUS como variable disponible para la comunicación entre el programa de aplicación y el DBMS, y se supone que un valor de cero para DB_STATUS indica una operación satisfactoria. Las actualizaciones de los registros se practican usando STORE, ERASE y MODIFY, mientras que acciones como añadir o eliminar miembros de los conjuntos o transferir un registro de una ocurrencia de conjunto a otra se ejecutan a través de los mandatos CONNECT, DISCONNECT y RECONNECT. Los verbos/mandatos reales que usan diferentes sistemas de redes pueden diferir de los que se muestran en la tabla. Operaciones como las agregaciones se realizan usando los recursos del lenguaje de programación anfitrión.

Obsérvese que los mandatos se refieren al concepto de **actualidad** en el modelo de redes. El sistema mantiene un indicador «actual» para cada tipo de registros y tipo de conjuntos en el esquema. El valor del indicador de actualidad para un tipo de registros es un puntero a la más reciente ocurrencia de registro para ese tipo de registros. El valor del indicador de actualidad para un tipo de conjuntos es un puntero a la más reciente ocurrencia de registro para un tipo de registros que sea miembro o bien propietario de ese conjunto.

Para navegar a través de la base de datos de redes, hay disponibles distintas formas del mandato FIND.

- **FIND ANY <nombre de tipo de registros> [USING <lista de campos>]**
Este se usa para encontrar un registro con base en un conjunto de condiciones de igualdad respecto a uno o más campos dentro del registro.
- **FIND DUPLICATE <nombre tipo de registros> [USING <lista de campos>]**
Este se usa para encontrar un registro que tiene los mismos valores en uno o más de sus campos que el registro actual.
- **FIND (FIRST|NEXT|PRIOR|LAST|...) <nombre tipo de registros> WITHIN <nombre tipo de conjuntos> [USING <nombres de campos>]**
Este se usa para «moverse dentro de un conjunto» desde el registro propietario para encontrar un registro miembro (primero [FIRST], último [LAST] o enésimo, según se especifique) que coincida con un conjunto de condiciones de igualdad respecto a uno o más campos dentro del registro. Se usa también para «moverse dentro de un conjunto» desde un registro miembro al próximo registro miembro (siguiente [NEXT] o anterior [PRIOR], según se especifique) que corresponda a un conjunto de condiciones de igualdad respecto a uno o más campos dentro del registro.
- **FIND OWNER WITHIN <nombre tipo de conjuntos>**
Este se usa para «moverse dentro de un conjunto» desde un registro miembro para encontrar el único registro propietario del cual es miembro en la ocurrencia de conjunto actual.

La figura 13.10 muestra la especificación de navegación de una operación



Consulta 1: Encontrar los instructores que tengan menos de 30 años de edad



Especificación de navegación en ER

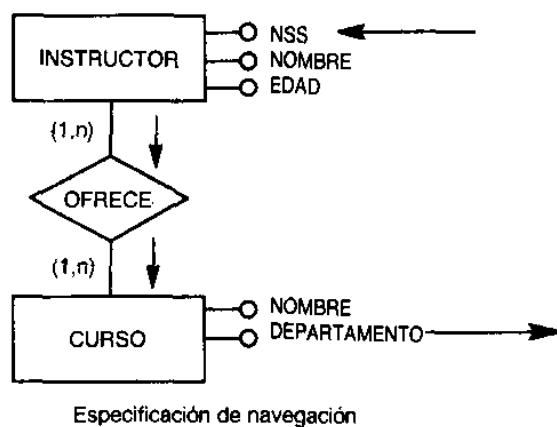
```
%FIND ANY INSTRUCTOR
mientras DB_STATUS=0 hacer
  comenzar
    %GET INSTRUCTOR
    si EDAD < 30 entonces
      comenzar
        escribir información de instructor
      terminar
    %FIND ANY INSTRUCTOR
  terminar
```

Especificación de DML de redes

Figura 13.10. Esquema de navegación en ER y especificación en DML de redes de una recuperación condicional de un tipo de entidades.

simple con la entidad **INSTRUCTOR**. Se muestran los mandatos de DML de redes precedidas de un signo de %. Obsérvese que la condición «menos de 30 años de edad» corresponde a una selección por **EDAD** que *no puede* ser usada directamente en ninguna forma del mandato **FIND** anterior. En consecuencia, nos vemos forzados a realizar una búsqueda exhaustiva a través de todos los instructores para encontrar aquellos que cumplen la condición requerida.

En la figura 13.11, el esquema de navegación requiere el acceso a **CURSO** con una navegación de **INSTRUCTOR** a **CURSO** a través de **OFRECE**. Esto se traduce en un **FIND** para el **INSTRUCTOR** usando el valor de **NSS** dado, seguido por una recuperación de los registros miembros del conjunto **OFRECE**. Obsérvese que los



```

INSTRUCTOR.NSS:=$NSS
%FIND ANY INSTRUCTOR USING NSS
si DB_STATUS= 0 entonces
    comenzar
        %FIND FIRST CURSO WITHIN OFRECE
    mientras DB_STATUS=0 hacer
        comenzar
            %GET CURSO
            escribir CURSO DEPARTAMENTO
            %FIND NEXT CURSO WITHIN OFRECE
        terminar
    terminar
    
```

Especificación en DML de redes

Figura 13.11. Esquema de navegación en ER y especificación en DML de redes de una operación de recuperación desde múltiples tipos de entidades.

ciclos a través de todos los cursos que son miembros de la ocurrencia (actual) del conjunto son controlados por el programa anfitrión³.

13.4. Base de datos del caso de estudio en el modelo de redes

Ahora se procede con el caso de estudio de los capítulos anteriores. Se aplica la metodología que se mostró en el apartado 13.2 para generar el esquema de redes lógico. Luego se muestra la transformación de algunos ejemplos de operaciones de navegación al DML de redes, con base en el material del apartado 13.3.

13.4.1. Correspondencia de esquemas con el modelo de redes

La correspondencia del esquema conceptual a lógico de la figura 11.15 con un esquema de redes es bastante sencilla. Se empieza por considerar el modelado

³ Los bucles y las estructuras selectivas se han traducido a pseudocódigo.

de identificaciones externas. Se resuelve la identificación externa de SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP provista por la entidad VIAJE mediante la inclusión del atributo NUMERO_VIAJE en los dos tipos de registros, SEGMENTO_DE_RECORRIDO_ORD y TIENE_SEG_ORD. Obsérvese que los tipos de conjuntos TIENE_SEG_POP y TIENE_SEG_ORD se encargan implícitamente de esta identificación externa. No obstante, se elige incluir el identificador externo explícitamente en los dos tipos de registros antes citados para que no se requiera un acceso a VIAJE nada más para obtener NUMERO_VIAJE.

SEGMENTO_DE_RECORRIDO_DIARIO es un tipo de registros usado muy frecuentemente, como se vio en el capítulo 11; así que también se muestra con la clave entera, el trío (NUMERO_VIAJE, NUMERO_SEGMENTO, FECHA) y no sólo con FECHA. No hay atributos polivalentes o compuestos. Cada entidad se corresponde con un tipo de registros. Los tipos de entidades denominados SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP se corresponden con los tipos de registros llamados SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP en el esquema de redes.

Finalmente, hay que hacer corresponder las interrelaciones restantes con conjuntos. Las interrelaciones de uno a muchos ORD_EN y POP_EN entre el tipo de entidades VIAJE, por un lado, y los tipos de entidades SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP, por el otro, corresponden respectivamente con los tipos de conjuntos TIENE_SEG_ORD y TIENE_SEG_POP, respectivamente, en el esquema de redes, con el tipo de registros VIAJE como propietario. Las interrelaciones de uno a muchos DE_S_R_ORD y DE_S_R_POP entre los tipos de entidades SEGMENTO_DE_RECORRIDO_ORD y SEGMENTO_DE_RECORRIDO_POP, por una parte, y el tipo de entidades SEGMENTO_DE_RECORRIDO_DIARIO, por la otra, corresponden con dos tipos de conjuntos, DE_ORD y DE_POP, incidentes en el tipo de registros SEGMENTO_RECORRIDO_DIARIO del esquema de redes. Las dos interrelaciones de muchos a muchos, TIENE_RESERVA y ESTA_A_BORDO corresponden con registros vínculo nuevos con nombres idénticos; ambos están conectados por dos tipos de conjuntos a sus tipos de registros propietarios respectivos. La figura 13.12 muestra el esquema de redes final⁴.

13.4.2. Correspondencia de operaciones con el modelo de redes

En la figura 13.13 se muestra un ejemplo de correspondencia de tres consultas de navegación de la base de datos del caso de estudio con su correspondiente representación en DML de redes. Para la operación de recuperación O2, las ciudades de salida y llegada se introducen al programa de aplicación y se almacenan en variables del programa. Usándolas, se hace una primera búsqueda del segmento de recorrido en el tipo de registros SEGMENTO_DE_RECORRIDO_POP. Si no tiene éxito, la búsqueda continúa en el tipo de registros SEGMENTO_DE_RECORRIDO_ORD. Se ilustra una operación de inserción para crear un cliente nuevo

⁴ Sólo la parte pertinente marcada del esquema ER de la figura 11.15 se ha hecho corresponder con el esquema de redes.

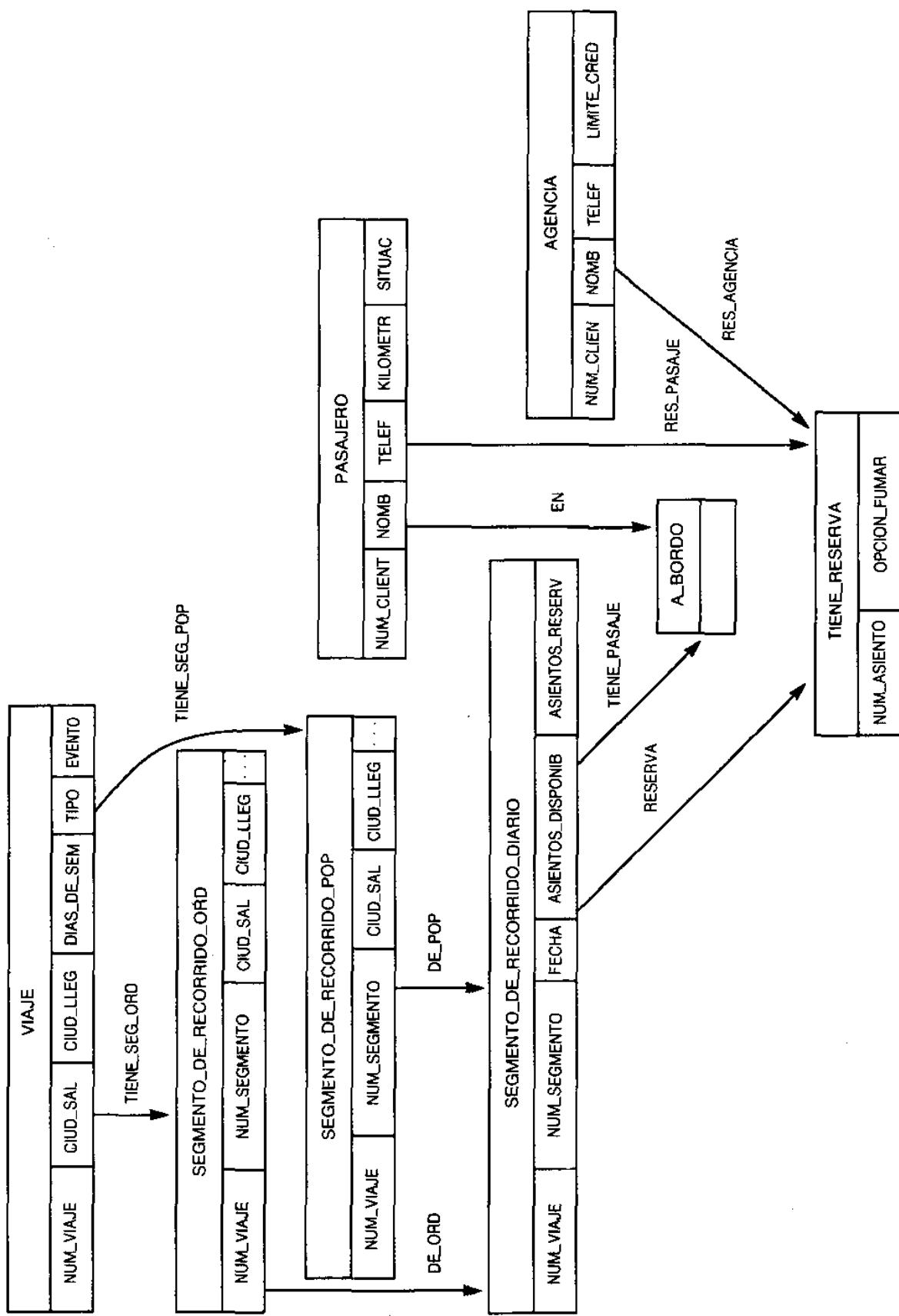


Figura 13.12. El esquema lógico de redes del caso de estudio.

Operación O2: Buscar un segmento de recorrido por ciudad de salida y ciudad de llegada (véase el esquema de navegación de la figura 10.11).

SR indica una variable de buffer en el área de trabajo del usuario, con el mismo tipo que los segmentos de recorrido.

```

leer($S,$LL)
SR.CIUDAD_SALIDA:=$S
SR.CIUDAD_LLEGADA:=$LL
mientras DB_STATUS=0 hacer
    comenzar
        %FIND ANY SEGMENTO_DE_RECORRIDO_POP USING SR.CIUDAD_SALIDA, SR.CIUDAD_LLEGADA
        si DB_STATUS=0 entonces
            comenzar
                %GET SEGMENTO_DE_RECORRIDO_POP
                escribir SR.NUM_VIAJE, SR.NUM_SEGMENTO, SR.HORA_SALIDA, SR.HORA_LLEGADA
                terminar
        si no
            comenzar
                %FIND ANY SEGMENTO_DE_RECORRIDO_ORD USING SR.CIUDAD_SALIDA, SR.CIUDAD_LLEGADA
                si DB_STATUS=0 entonces
                    comenzar
                        %GET SEGMENTO_DE_RECORRIDO_ORD
                        escribir SR.NUM_VIAJE, SR.NUM_SEGMENTO, SR.HORA_SALIDA, SR.HORA_LLEGADA
                        terminar
                    terminar
                terminar
            terminar
        terminar
    terminar

```

Operación O4: Crear un cliente nuevo (véase el esquema de navegación de la figura 10.12)

Aquí \$NUMC, \$NOMBRE, \$TEL, son valores de entrada apropiados para un cliente. \$KIL, \$SIT se aplican a un pasajero individual, mientras que \$LCREDITO se aplica a una agencia. \$TIPO.CLIENTE = «PAS» o \$TIPO.CLIENTE = «AGE» determinan si el cliente es un pasajero o una agencia. Se supone que las variables de área de trabajo del usuario para CLIENTE, PASAJERO y AGENCIA tienen los mismos nombres.

```

leer($TIPO.CLIENTE)
si $TIPO.CLIENTE='PAS' entonces
    comenzar
        leer($NUMC, $NOMBRE, $TEL, $KIL, $SIT)
        PASAJERO.NUM_CLIENTE:=$NUMC
        PASAJERO.NOMB:=$NOMBRE
        PASAJERO.TELEF:=$TEL
        PASAJERO.KILOMETR:=$KIL
        PASAJERO.SITUAC:=$SIT
        STORE PASAJERO
    terminar
    si-no
        comenzar
            leer($NUMC, $NOMBRE, $TEL, $LCREDITO)
            AGENCIA.NUM_CLIENTE:=$NUMC
            AGENCIA.NOMB:=$NOMBRE
            AGENCIA.TELEF:=$TEL
            AGENCIA.LIMITE_CRED:=$LCRED
            STORE AGENCIA
        terminar;

```

Figura 13.13. Correspondencia de algunos ejemplos de esquemas de navegación con DML de redes.

Operación O6: Borrar reservas de un viaje pasado (véase el esquema de navegación de la figura 10.14).

V, SR, SRD, indican variables de buffer en el área de trabajo del usuario, con los mismos tipos que las entidades viaje, segmento de recorrido y segmento de recorrido diario, respectivamente

```

leer($V, $S, $F)
V.NUM_VIAJE.=$V
SR.NUM_SEGMENTO.=$S
SRD.FECHA.=$F
%FIND ANY VIAJE USING V.NUM_VIAJE
si DB_STATUS=0 entonces
    comenzar
        %FIND NEXT SEGMENTO_DE_RECORRIDO_POP WITHIN TIENE_SEG_POP USING SR.NUM_SEGMENTO
        si DB_STATUS=0 entonces
            comenzar
                %FIND NEXT SEGMENTO_DE_RECORRIDO_DIARIO WITHIN DE_POP USING SRD.FECHA
                mientras DB_STATUS=0 hacer
                    comenzar
                        FIND NEXT TIENE_RESERVA WITHIN RES
                        %ERASE TIENE_RESERVA
                    terminar
                terminar
            terminar
        si no
            FIND NEXT SEGMENTO_DE_RECORRIDO_ORD WITHIN TIENE_SEG_ORD USING SR.NUM_SEGMENTO
            si DB_STATUS=0 entonces
                comenzar
                    FIND NEXT SEGMENTO_DE_RECORRIDO_DIARIO WITHIN DE_ORD USING SRD.FECHA
                    mientras DB_STATUS=0 hacer
                        comenzar
                            %FIND NEXT TIENE_RESERVA within RES
                            %ERASE TIENE_RESERVA
                        terminar
                    terminar
                terminar
            terminar
        terminar
    terminar

```

Figura 13.13.Bis Correspondencia de algunos ejemplos de esquemas de navegación con DML de redes (*continuación*).

(operación O4). Dependiendo del tipo de cliente («PAS» o «AGE») se crea un registro de PASAJERO o AGENCIA. Finalmente, la eliminación de una reserva existente se muestra en la operación O6. Antes de borrar el registro TIENE_RESERVA con un mandato DELETE, se averigua si el viaje atañe a un SEGMENTO_DE_RECORRIDO_ORD o a un SEGMENTO_DE_RECORRIDO_POP. Obsérvese que los registros TIENE_RESERVA por borrar pueden ser propiedad de agencias o de pasajeros.

13.5. Retroingeniería de los esquemas de redes a esquemas ER

En el apartado 12.5 se introdujo el concepto de retroingeniería y se apuntaba que quizás se necesitaría establecer correspondencias entre bases de datos representadas en modelos lógicos y bases de datos representadas en modelos de más alto nivel, más abstractos (como el modelo ER), por tres razones principales:

1. Para obtener un buen entendimiento conceptual de los datos almacenados bajo los diversos sistemas en las organizaciones.
2. Para *repetir* el diseño físico y lógico de la base de datos, de manera que satisfaga las necesidades de las aplicaciones actuales.
3. Para convertir las bases de datos de redes existentes en bases de datos relacionales (o bases de datos orientadas a objetos, aunque no se tratan aquí) usando primero el diseño conceptual ER como diseño intermedio. Luego se puede aplicar la metodología de diseño lógico del capítulo 12.

Muchas organizaciones han implantado bases de datos que usan los modelos dominantes anteriores (los jerárquicos y de redes, y sus variantes), según las versiones de los distintos proveedores. Miles de instalaciones que usan bases de datos implantadas según el modelo de redes siguen trabajando hoy día. En consecuencia, el problema de retroingeniería es en realidad más crucial en el caso de estos dos modelos que en el del modelo relacional, comentado ampliamente en el apartado 12.5. El modelo de redes tuvo el más amplio seguimiento en la industria; los proveedores de los años setenta ofrecieron productos como IDMS (de Cullinet, ahora de Computer Associates), IDS II (de Honeywell), VAX-DBMS (de Digital), DMS 1100 (de Univac) e IMAGE (de Hewlett-Packard), para mencionar los más importantes. Estas bases de datos han sido desarrolladas por muchos individuos y han evolucionado durante largo tiempo. Por eso es importante desarrollar técnicas para captarlas en forma abstracta con miras a cualquier desarrollo posterior, como en los casos indicados en los puntos 2 y 3 anteriores.

En este apartado se trata el problema de abstraer un esquema de base de datos de redes dado para obtener su correspondiente esquema ER. Esta correspondencia es mucho más sencilla que la del modelo relacional porque ambos modelos, el de redes y el ER, representan a los esquemas empleando una estructura gráfica similar formada por nodos y enlaces. Los nodos representan a las entidades, y los enlaces a las interrelaciones. La información adicional, como las restricciones de cardinalidad, tendría que ser desarrollada con base en lo que sepa el administrador de la base de datos, o los usuarios/diseñadores.

13.5.1. Una metodología para establecer la correspondencia de un esquema de redes con un esquema ER

Supongamos que los tipos de conjuntos multimiembros no se permiten en el esquema de redes⁵. El procedimiento general consiste en los siguientes pasos:

Paso 1: Hacer corresponder registros con entidades. Para cada tipo de registros, definir un tipo de entidades. La clave completa del registro (posible-

⁵ Si se permiten conjuntos recursivos, éstos se pueden transformar usando anillos o interrelaciones recursivas; véase el apartado 2.2. Los conjuntos multimiembros se pueden tratar transformando cada miembro en una interrelación individual. Sin embargo, esto distorsiona la intención original.

mente después de una propagación de la clave del registro propietario al miembro) se convierte en el identificador completo de la identidad.

Paso 2: Tratamiento especial de ciertas entidades. Este paso implica el postprocesamiento de ciertas entidades después del primer paso. Algunas veces el esquema de redes usa registros vínculo sin atributos para modelar interrelaciones de muchos a muchos. Las entidades correspondientes que resulten del paso anterior deberán ser reemplazadas por interrelaciones de muchos a muchos en el modelo ER.

Algunas veces el esquema de redes usa registros ficticios o incluso registros con nombre para representar interrelaciones recursivas. Las entidades correspondientes que resulten de esa estrategia deben integrarse en los tipos de entidades principales o dominantes de los cuales surjan. Si el registro ficticio tiene atributos que describan la función de la entidad ficticia, se debe transferir dichos atributos a la entidad principal. Por ejemplo, supóngase que se nos hubiera dado el esquema de redes de la figura 13.9b. En el paso 1, el registro FICTICIO se traduciría a un tipo de entidad, el cual se podría integrar de nuevo en el tipo de entidades EMPLEADO que el registro FICTICIO representa. Los dos tipos de conjuntos se convierten en un tipo de interrelaciones con las restricciones de cardinalidad apropiadas.

Si un registro vínculo se usa con atributos para modelar una interrelación de muchos a muchos, los atributos pertenecen generalmente a la interrelación que está siendo representada. Se debe identificar tales registros y luego eliminar las entidades correspondientes resultado del paso 1, sustituyéndolas por una interrelación con los atributos apropiados.

Paso 3: Representar cada tipo de registros por una interrelación de uno a muchos. En general, cada tipo de conjuntos en el modelo de redes representa una interrelación. Inicialmente, haremos corresponder cada uno con un tipo de interrelación.

Paso 4: Tratamiento especial de ciertas interrelaciones de uno a muchos. Dos tipos de interrelaciones que resultan del paso 3 necesitan tratamiento especial:

1. Las interrelaciones cuyo propósito es transferir un identificador externo de un tipo de entidades al otro. Esta identificación externa se muestra explícitamente en el esquema ER.
2. Las interrelaciones ES_UN, que pueden representar semánticamente generalizaciones o subconjuntos, deben ser sustituidas mediante la creación de una interrelación de subconjunto entre las entidades implicadas, creando una jerarquía de generalización. Se debe establecer las especificaciones de total o parcial y superpuesta o exclusiva de acuerdo con el conocimiento de las entidades participantes.

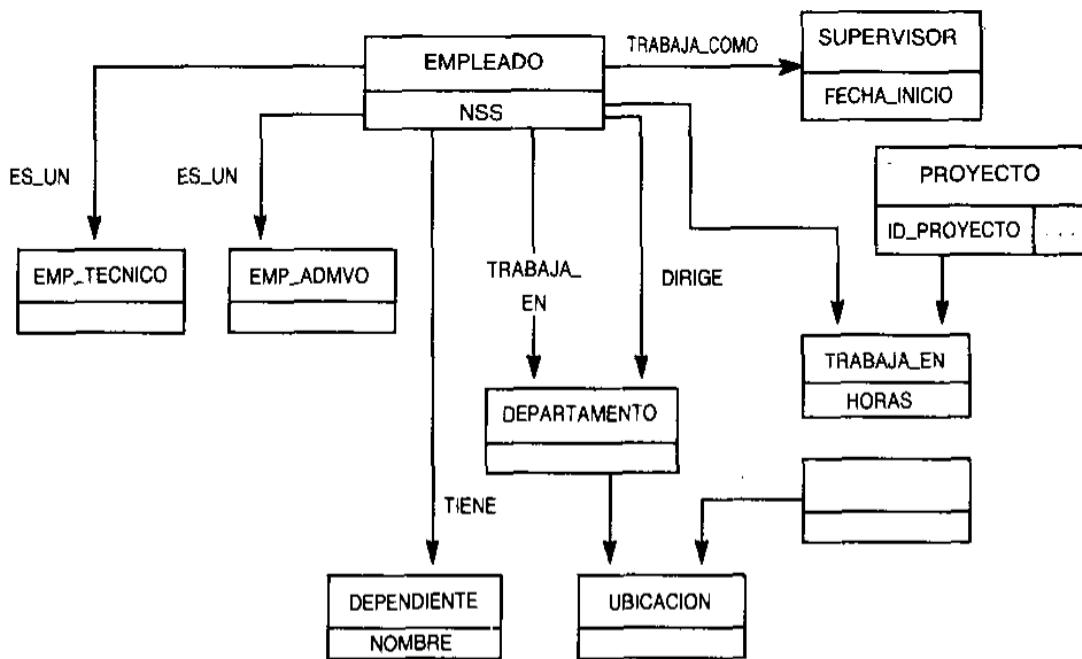


Figura 13.14. Un esquema de redes.

Paso 5: Tratar los conjuntos propiedad de sistema y multimiembros. Obsérvese que los conjuntos propiedad de sistema *no pueden* transformarse al modelo ER. De manera similar, los conjuntos multimiembros no pueden representarse sin alguna modificación.

Finalmente, téngase en cuenta que los identificadores deben ser asignados manualmente en el esquema ER porque el esquema de redes (en su representación diagramática) carece de una definición de claves.

13.5.2. Ejemplo de correspondencia de un esquema de red con un esquema ER

Considérese el esquema de redes de la figura 13.14. Para ilustrar la metodología antes citada, se han introducido tipos de conjuntos que tienen como resultado diversas construcciones del modelado ER después de la correspondencia. En el paso 1 todos los registros se hacen corresponder con entidades. En el paso 2 la «entidad vacía» que corresponde al registro UBICACION se elimina, sustituyéndola por una interrelación de muchos a muchos entre DEPARTAMENTO y EDIFICIO. De manera similar, la entidad TRABAJA_EN, que representa la interrelación de muchos a muchos entre EMPLEADO y PROYECTO, se convierte en una interrelación con HORAS como atributo. SUPERVISOR es una entidad que representa otro papel de la entidad dominante EMPLEADO; la representamos como la interrelación recursiva SUPERVISION y transferimos el atributo FECHA_INICIO a esta interrelación. En el paso 3, para los tipos de entidades que quedan en el

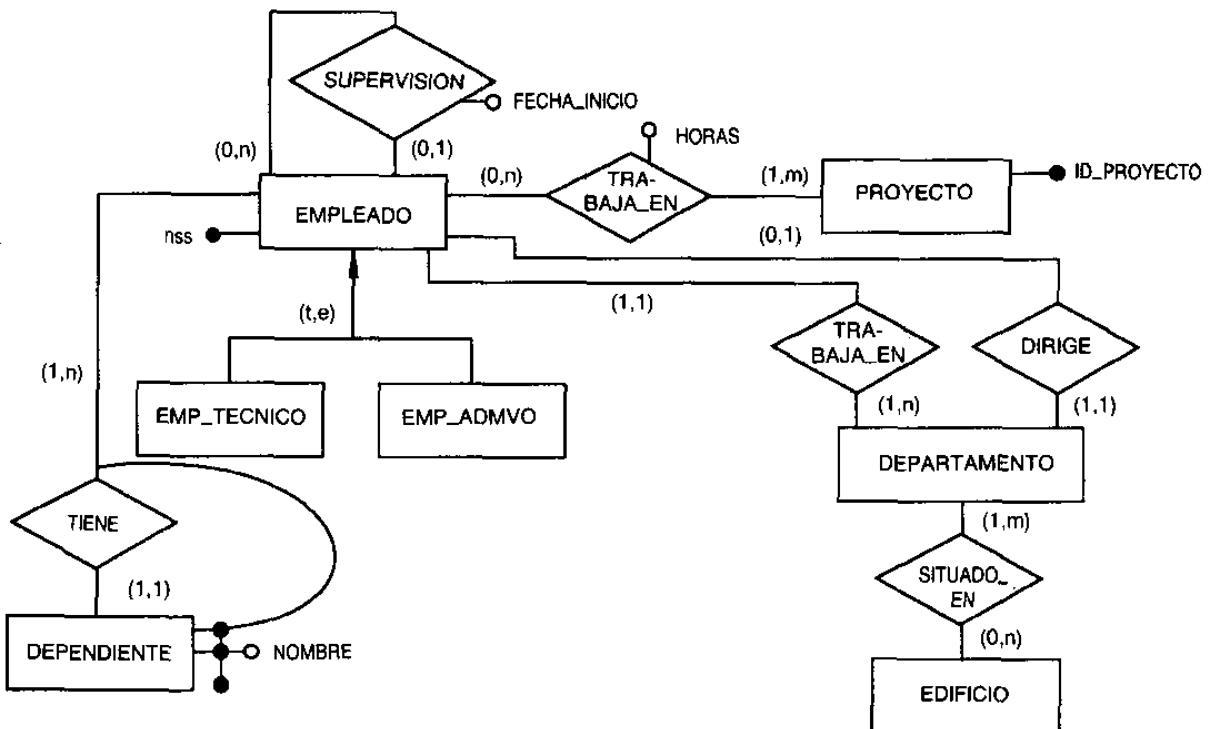


Figura 13.15. Correspondencia del esquema de redes en un esquema ER.

esquema, todos los conjuntos pertinentes del esquema de redes se convierten en interrelaciones de uno a muchos. En este paso, el tipo de conjuntos TIENE se hace corresponder con la interrelación TIENE en el esquema ER. En el paso 4 se descubre que el propósito de la interrelación TIENE es transferir el identificador externo NSS del tipo de entidades EMPLEADO al tipo de entidades DEPENDIENTE, para así poder construir un identificador mixto (NSS, NOMBRE) para el tipo de entidades DEPENDIENTE. Se modela esta situación explícitamente en el esquema ER mostrando DEPENDIENTE como identificado externamente por EMPLEADO. Por último, EMP_TECNICO (empleados técnicos) y EMP_ADMIN (empleados administrativos) se reconocen como subconjuntos de la entidad EMPLEADO, y se crea la jerarquía de generalización correspondiente. El conocimiento externo de estos dos tipos de entidades sugiere una especificación total y exclusiva para esta jerarquía. El resultado global de la correspondencia se muestra en la figura 13.15.

13.6. Resumen

En este capítulo se introdujeron los conceptos básicos que fundamentan el modelo de datos de redes, incluidas las nociones de tipos de registros, atributos y

tipos de conjuntos. Estos están de acuerdo con el informe DBTG de CODASYL y sus posteriores versiones, aunque no se sigue estrictamente ninguna versión en particular de este informe. Se presentó el conjunto de rasgos de uso más común en los sistemas implantados actualmente. A diferencia del modelo relacional, el modelo de redes permite vectores y grupos de repetición, que facilitan emplear atributos polivalentes y compuestos.

Se expuso la noción de ocurrencia de conjunto y diversas especificaciones que se necesitan para cada conjunto, incluidos el ordenamiento de conjunto, la selección de conjunto y la modalidad de conjunto. Las opciones de inserción y eliminación permiten especificar restricciones adicionales respecto a la pertenencia a conjuntos. Entre los tipos de conjuntos especiales están los conjuntos propiedad del sistema y los multimiembros. Los últimos se permiten en muy pocos sistemas. La noción de claves no está bien definida en el modelo de redes; los registros no necesitan claves. Las restricciones de integridad en el modelo de redes se presentaron en forma de restricciones sobre los conjuntos.

Se dio una metodología general para transformar esquemas ER en esquemas de redes considerando la correspondencia de atributos compuestos y polivalentes, identificadores externos y entidades. Las interrelaciones corresponden con tipos de conjuntos, excepto las de muchos a muchos, para las cuales se debe establecer un tipo de registros aparte denominado *registro vínculo*. La correspondencia de interrelaciones se explicó individualmente para las de uno a uno, de uno a muchos, muchos a muchos, n-arias y recursivas. Se dio un procedimiento general para transformar operaciones a partir de esquemas de navegación ER al lenguaje de manipulación de datos de redes. Se mostró la correspondencia de tres esquemas de navegación representativos que implican una búsqueda, una inserción y una eliminación (del capítulo 10) con el lenguaje de manipulación de datos de redes para el caso de estudio. Es interesante observar que el procedimiento del diseño lógico aquí delineado tiene varios rasgos en común con el diseño lógico de bases de datos orientadas a objetos, que también tiene esquemas con estructura esencialmente de gráfica.

Por último, se trató el problema de la retroingeniería desde una base de datos de redes dada a un esquema conceptual abstracto en el modelo ER. Esto es muy importante para organizaciones donde las bases de datos de redes han estado en uso durante un periodo largo, o donde los datos se han convertido a sistemas de redes sin un diseño adecuado de bases de datos. En la mayoría de las organizaciones, un entendimiento conceptual de las bases de datos existentes puede ser muy útil antes de implantar nuevas aplicaciones o convertir esas bases de datos en DBMS más nuevos, como los relacionales. Los esquemas conceptuales ER pueden servir como representación intermedia. Se proporcionó un procedimiento general, paso por paso, para llegar al esquema conceptual. Esto es mucho más directo que el procedimiento correspondiente para el modelo relacional; se apoya en la capacidad del diseñador para detectar información adicional, una parte de la cual es explícita (como los registros vínculo, que captan interrelaciones de muchos a muchos) y otra implícita, como las interrelaciones conjunto-subconjunto, las interrelaciones recursivas o las restricciones de car-

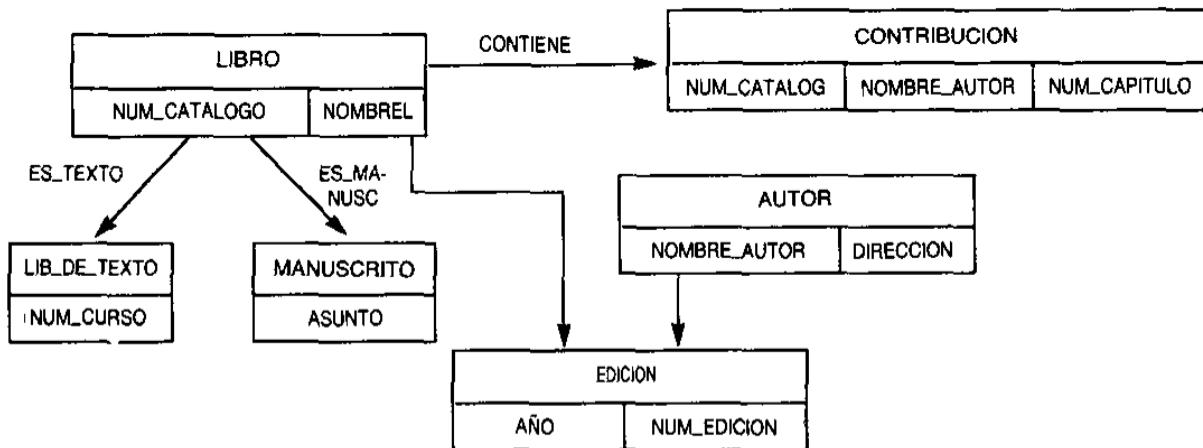


Figura 13.16. Esquema de redes de la base de datos de una biblioteca.

dinalidad. Esta información provee la semántica adicional que se necesita para construir un esquema conceptual rico en el modelo ER.

Ejercicios

- 13.1. Considere la base de datos de proyectos de investigación de la figura 2.35. Aplique la metodología descrita en este capítulo y transforme el esquema lógico ER en un esquema de redes.
- 13.2. Considere la base de datos de universitaria de la figura 2.33. Sustituya la jerarquía de generalización introduciendo interrelaciones entre las entidades superconjunto y subconjunto. Aplique la metodología descrita en este capítulo y transforme el esquema lógico ER en un esquema de redes.
- 13.3. Repita el ejercicio 2 con la base de datos de fútbol de la figura 2.34.
- 13.4. Considere el diagrama ER de la base de datos para los departamentos de tráfico y parques de una ciudad (Fig. 12.21).
 - a) Dibuje esquemas de navegación ER para las siguientes consultas (esta parte es igual al ejercicio 12.4):
 1. Liste todos los nombres de las calles de una ciudad dada.
 2. Liste todas las intersecciones de la calle Main en la ciudad de Gainesville, Georgia.
 3. Liste todos los parques de la ciudad de Gainesville, Florida.
 4. Liste todos los parques localizados en la calle Huron de Ann Arbor, Michigan.
 - b) Convierta primero el diagrama ER en una base de datos de redes.
 - c) Convierta los esquemas de navegación anteriores al lenguaje de manipulación de datos de algún DBMS de redes que usted conozca.

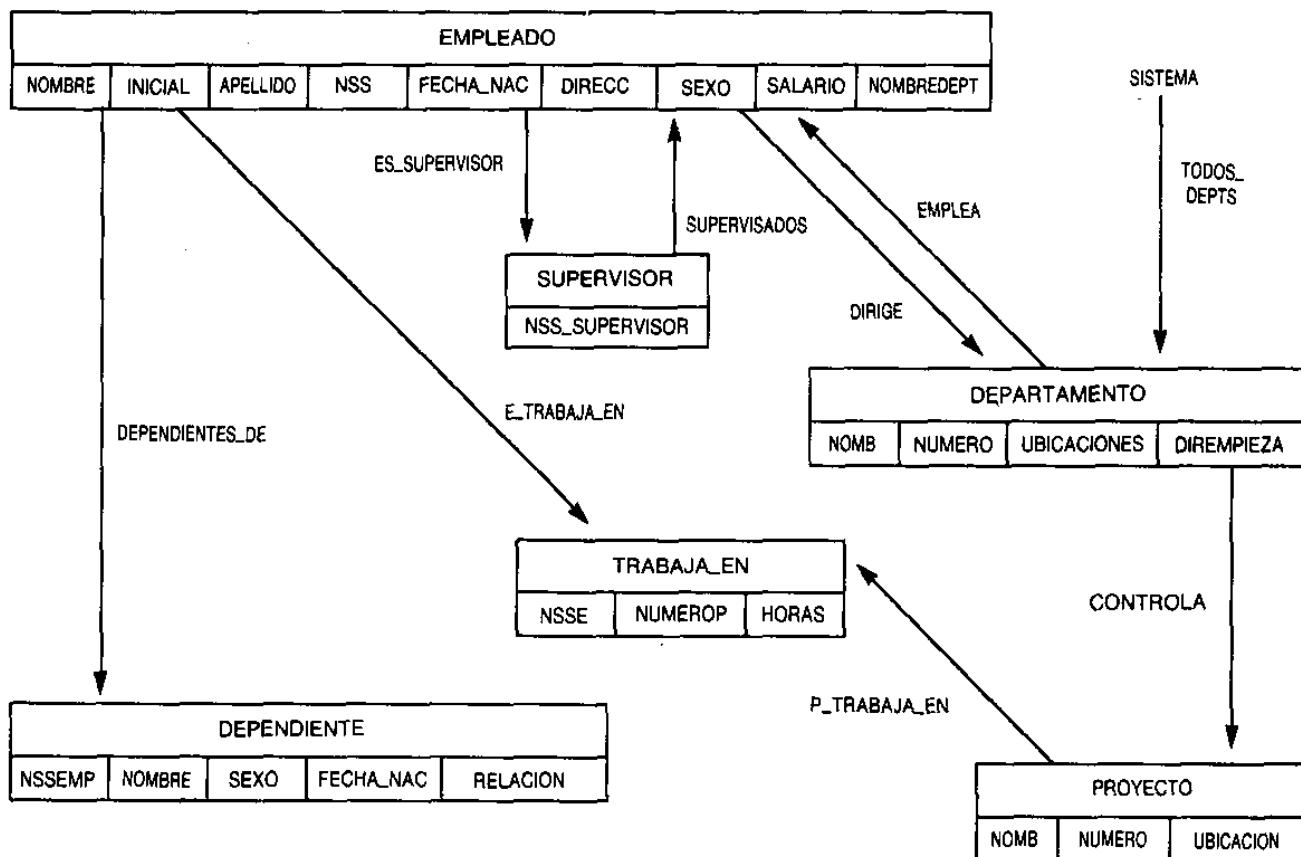


Figura 13.17. Esquema de redes de la base de datos de una compañía.

- 13.5. Considere el esquema ER de la base de datos de aeropuerto y vuelos de la figura 12.22. Conviértalo en un esquema de redes. Haga las suposiciones necesarias durante la correspondencia y especifíquelas claramente.
- 13.6. Considere el esquema ER al que se llegó en la figura 13.15. Convierta la jerarquía de generalización introduciendo interrelaciones ES_UN. Con su conocimiento de este ejemplo, aplique el procedimiento de correspondencia ER-redes y verifique si su resultado coincide con el esquema con que se inició en la figura 13.14
- 13.7. Considere el esquema de redes de la base de datos de una biblioteca que se muestra en la figura 13.16
La base de datos se explica por sí misma . Aplicando la metodología del apartado 13.5, haga corresponder el esquema anterior con un esquema ER; tome nota de que los libros de texto y los manuscritos son subclases de libros. Verifique que sus resultados coincidan con los del ejercicio 12.7. Si no es así, determine las razones de las discrepancias.
- 13.8. Considere la base de datos mantenida por el departamento de

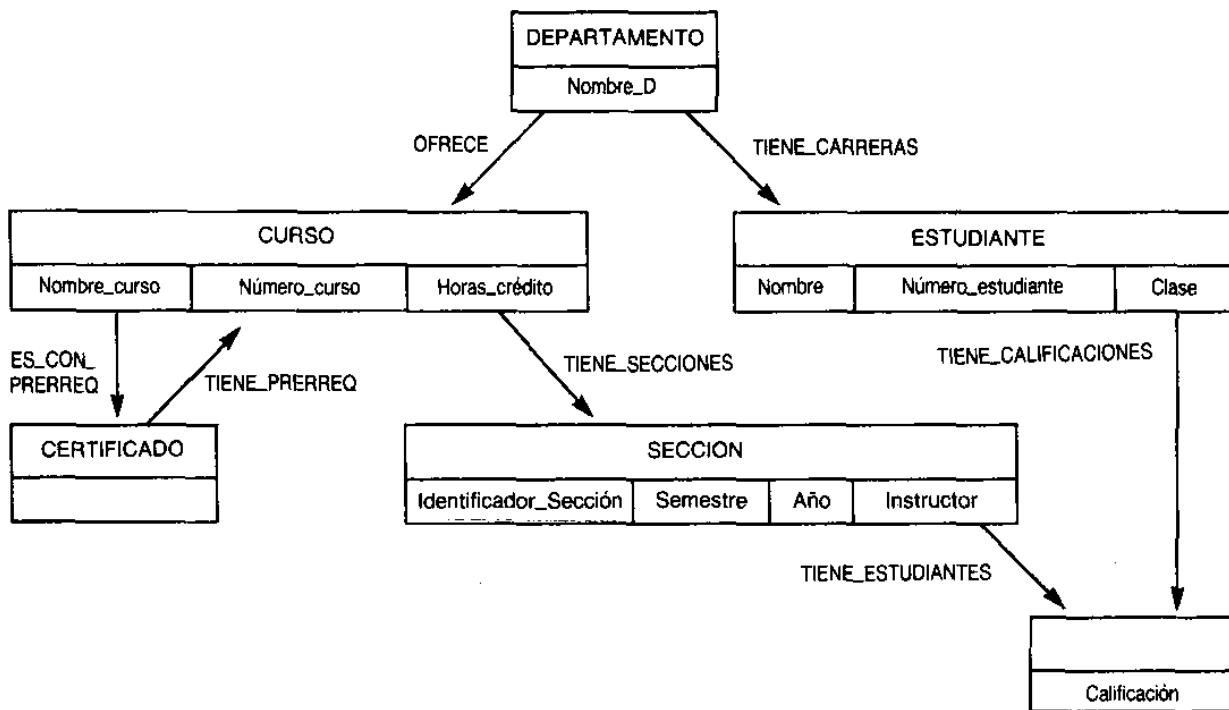


Figura 13.18. Esquema de redes de la base de datos de una universidad.

personal de una compañía (Fig. 13.17). El significado de la base de datos es obvio. NSS en EMPLEADO, NSSEMP en DEPENDIENTE, y NSSEMP en TRABAJA_EN representan el NSS del empleado, que es la clave de un empleado. NSS_SUPERVISOR es lo mismo que NSS del empleado. Haga cualquier suposición adicional que necesite. Utilice la metodología del apartado 13.5 para hacer corresponder el esquema anterior con un diagrama ER, y compare este último con el resultado del ejercicio 12.8. En caso de haber diferencias, determine la causa.

- 13.9. Partiendo de la base de datos de redes de la figura 13.8, aplique el procedimiento de correspondencia inversa para convertir este esquema en un esquema ER. Compare el resultado con el esquema ER de la base de datos de control de pedidos de la figura 13.7.
- 13.10. Partiendo de la base de datos de redes de la figura 13.12, y con su conocimiento del caso de estudio, aplique correspondencia inversa para convertir éste en un esquema ER. Compare el resultado con el esquema ER al que se llegó en la figura 11.15.
- 13.11. Considere la base de datos de redes de la figura 13.18, que corresponde a una universidad.

El significado de la base de datos es obvio. Un curso puede pertenecer a un solo departamento y tener múltiples requisitos. Los cursos

se imparten en múltiples secciones. Cada estudiante recibe una calificación para la sección en la cual se ha inscrito. Haga cualquier suposición adicional que necesite y utilice la metodología del apartado 13.5 para hacer corresponder el esquema anterior con un esquema ER.

Bibliografía

- C. Bachman, «The Data Structure Diagrams», *Data Base (Boletín de la ACM-SIGFILET)*, 1, núm. 2, marzo de 1969.
- C. Bachman, «The Data Structure Set Model». En R. Rustin, ed., *Proc. ACM-SIGMOD Debate on Data Models: Data Structure Set versus Relational*, 1974.
- C. Bachman y S. Williams, «A General-Purpose Programming System for Random Access Memories», En *Proc. Fall Joint Computer Conference, AFIPS*, 26, 1964.

Estos artículos representan los primeros trabajos de Bachman sobre el modelo de redes. El primero describe el concepto de diagramas de estructura de datos, que lleva a la noción de representar las estructuras de base de datos con diagramas de esquemas. El segundo artículo describe el primer DBMS de redes comercial, denominado IDS, desarrollado en GE/Honeywell.

Si se desea estudiar las características básicas del modelo de redes y del lenguaje de tratamiento de datos para este modelo, véanse los libros de Elmasri y Navathe, Date, Korth y Silberschatz y Ullman listados en la bibliografía del capítulo 1.

CODASYL, *Data Description Language Journal of Development*, Canadian Government Publishing Centre, 1978.

DBTG (Database Task Group de CODASYL), *The Database Task Group Report*, ACM, 1971.

Estas dos referencias dan principalmente la definición del modelo de redes como fue propuesto por el DBTG de CODASYL. El primer informe contiene lenguajes de descripción de datos (DDL) de esquemas y subesquemas, y un DML para usarse con Cobol. El segundo informe es una revisión del primero.

ANSI (American National Standards Institute), *The Database Language NDL*, ANSI documento X3.133, 1986.

Este documento trata sobre un estándar propuesto para el lenguaje de definición y tratamiento de redes preparado por el comité X3H2 de ANSI (Instituto Americano Nacional de Estándares) que todavía no ha sido aprobado formalmente.

S.R. Dumpala y S.K. Arora, «Schema Translation Using the Entity Relationship Approach». En P. Chen, ed., *Entity-Relationship Approach to Information Modeling and Analysis*, North-Holland, 1983.

Esta es la primera referencia que considera todos los casos de correspondencias: de ER a relacional, de redes y jerárquico, así como la correspondencia inversa de relacional, redes y jerárquico a ER. El enfoque está limitado al modelo ER básico y no es completo.

P. Bertaina, A. di Leva y P. Giolito, «Logical Design in CODASYL and Relational Environments». En S. Ceri, ed., *Methodology and Tools for Data Base Design*, North-Holland, 1983.

K. Irani, S. Purkayastha y T. Teorey, «A Designer for DBMS Processable Logical Design Structures», En A. Furtado, H. Morgan, eds., *Proc. International Conference on Very Large Databases*, Rio de Janeiro, 1979.

E. Wong y R. Katz, «Logical Design and Schema Conversion for Relational and DBTG Databases». En P. Chen, ed., *Entity-Relationship Approach to Systems Analysis and Design*, North-Holland, 1980, 311-22.

Los trabajos anteriores dan las normas generales para el diseño lógico de esquemas CODSASYL. El trabajo de Irani y colaboradores considera un diseño «óptimo» de un esquema de base de datos de redes, teniendo en cuenta los requerimientos de procesamiento del usuario.

S.B. Navathe, «An Intuitive View to Normalize Network Structured Data». En *Proc. International Conference on Very Large Databases*, Montreal, 1980.

Este trabajo expone el problema de correspondencia de un esquema de redes con un conjunto de interrelaciones. La propagación de identificadores está considerada en detalle para definir «caminos de identificación» dentro de una red, y describe un procedimiento simple para generar interrelaciones normalizadas.

Diseño lógico en el modelo jerárquico

El presente capítulo explica el tercero de los modelos de datos comerciales importantes: el modelo jerárquico. Este ha sido el modelo dominante en el mercado comercial debido a la aceptación del IMS (Information Management System, sistema de gestión de información) de IBM por un amplio segmento del mercado de procesamiento de datos. Este fue el primer DBMS jerárquico importante diseñado; posteriormente, System 2000 fue comercializado por MRI (ahora por SAS, Inc.) utilizando su propia versión del modelo jerárquico. A diferencia del modelo relacional, el jerárquico nunca fue definido formalmente. A diferencia del modelo de redes, el jerárquico no fue apoyado por una recomendación colectiva de lenguajes de DBMS jerárquicos, como en el informe del DBTG de CODASYL de 1971. Por tanto, no hay una definición estándar del modelo jerárquico. Aquí presentamos el modelo con nuestra propia terminología neutral y lo relacionamos con los recursos de IMS, que es el modelo jerárquico de DBMS dominante. Respecto a los recursos de modelado de datos, algunas veces se hace referencia específicamente a IMS. Por ejemplo, cuando se describe el lenguaje de manipulación de datos se siguen de cerca los recursos de IMS porque son ampliamente conocidos, pero se presentan usando una notación simple. La correspondencia de un esquema conceptual ER con un esquema jerárquico se trata de manera general. La correspondencia de operaciones usa un lenguaje objetivo que se parece a DL/1 de IMS.

En el apartado 14.1 se repasan los conceptos básicos y los recursos del modelo jerárquico. El apartado 14.2 expone la correspondencia de esquemas ER con el modelo jerárquico. Esto se efectúa haciéndolos corresponder primero con estructuras tipo redes y luego presentando un procedimiento para convertir esa estructura en jerarquías múltiples con registros duplicados; finalmente, se eliminan los duplicados usando interrelaciones virtuales. Después se consideran, de manera independiente, algunas situaciones especiales de modelado. Se demuestra la correspondencia con la base de datos de control de pedidos que se usó en el capítulo anterior. El apartado 14.3 expone la correspondencia de las especificaciones de esquemas de navegación con los recursos de manipulación de datos del modelo jerárquico. Las operaciones de manipulación de datos dis-

ponibles en el lenguaje DL/I de IMS se usan como ilustración. El apartado 14.4 muestra el diseño lógico de la base de datos del caso de estudio en el modelo jerárquico. Las mismas muestras de operaciones de navegación que se utilizaron en el apartado 13.4.2 se ilustran para este modelo. El apartado 14.5 presenta la retroingeniería de esquemas jerárquicos a esquemas ER.

14.1. El modelo jerárquico

En tanto que varios trabajos e informes forman la base de los modelos relacional y de redes, no hay una sola fuente o conjunto de documentos que defina el modelo jerárquico de datos. Los primeros sistemas de gestión de base de datos que se hicieron populares eran jerárquicos. El mercado comercial de DBMS estuvo dominado por el IMS de IBM desde el final de los años sesenta hasta principios de los años ochenta. Como resultado, un amplio número de bases de datos y aplicaciones que existen hoy día utilizan el enfoque jerárquico. Otro sistema popular es el System 2000 de SAS (originalmente comercializado por MRI). Hay varias diferencias entre estos dos sistemas respecto a los detalles del modelo de datos, los lenguajes de manipulación de datos y de consulta, y las estructuras de almacenamiento. Sin embargo, el enfoque esencial del modelado jerárquico es el mismo. En este apartado se presenta un análisis general del modelo jerárquico independiente de cualquier sistema específico.

El modelo jerárquico tiene dos conceptos básicos de estructuración: los tipos de registros y los tipos de interrelaciones padre-hijo. Un tipo de registros es la definición de un grupo de registros que almacenan información del mismo tipo. Un tipo de registros tiene muchas ocurrencias, denominadas **registros**. Cada tipo de registros contiene una colección de tipos de campos, que son elementos de datos con nombre. Cada tipo de campos está definido como un número entero, real, cadena de caracteres, etc., dependiendo de los tipos primitivos que trate el sistema. Un tipo de interrelaciones padre-hijo (**tipo IPH**) es una interrelación de uno a muchos entre un tipo de registros padre y un tipo de registros hijo.

Una ocurrencia de un tipo de interrelaciones padre-hijo consiste en un registro del *tipo de registros padre* y muchas (cero o más) ocurrencias del *tipo de registros hijo*. En adelante, se usará la palabra *registro* para designar el tipo o la ocurrencia, dependiendo del contexto, lo mismo se aplica a las interrelaciones padre-hijo, solamente cuando haya ambigüedad se usará específicamente el término *tipo de registros*, o *tipo de interrelaciones padre-hijo*.

Una esquema de base de datos jerárquica contiene varias jerarquías; cada una (o el esquema jerárquico completo) consiste en varios tipos de registros y tipos de IPH colocados de manera que formen un árbol. Considérese de nuevo la figura 13.1, que muestra un tipo de conjuntos y su ocurrencia en el modelo de redes. En el modelo jerárquico, ESTUDIANTE y CURSO serán dos tipos de registros, como antes, pero la interrelación padre-hijo con estudiante como padre

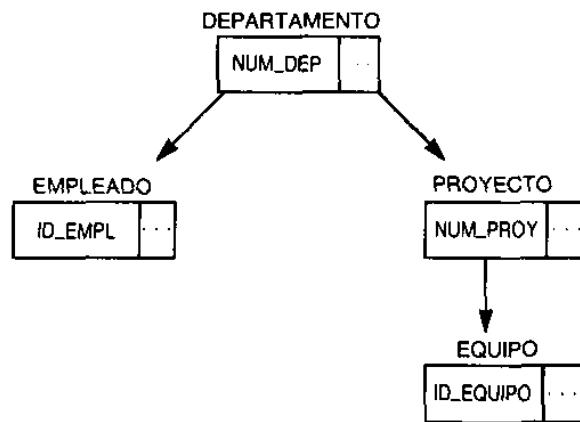


Figura 14.1. Un esquema jerárquico.

y curso como hijo no tendrá nombre. La ocurrencia de esta IPH será idéntica al conjunto de registros que se muestra en la figura 13.1.

La figura 14.1 muestra un esquema jerárquico con cuatro tipos de registros y tres tipos de IPH. Podemos referirnos a los tipos de IPH como el par (tipo de registros padre, tipo de registros hijo). Los tres tipos de IPH de la figura 14.1 son: (DEPARTAMENTO, EMPLEADO), (DEPARTAMENTO, PROYECTO) y (PROYECTO, EQUIPO). Aunque las IPH no tienen nombre, tienen asociado un significado; por ejemplo, en la figura 14.1, una ocurrencia del tipo de IPH (DEPARTAMENTO, PROYECTO) relaciona el registro padre de un departamento a los registros de los proyectos que controla. A diferencia del modelo de redes, el modelo jerárquico sólo puede tener una interrelación entre un par de tipos de registros; ésta es la razón por la que se puede dejar sin nombre.

14.1.1. Propiedades de los esquemas jerárquicos

Primero se definen dos términos: los tipos de registros *raíz* y los tipos de registros *hoja* en un esquema jerárquico. La *raíz* de la jerarquía es el registro más alto de la misma; no participa como tipo de registros hijo en ningún tipo de IPH. Un tipo de registros que no participa como padre en ningún tipo de IPH se denomina *hoja* de la jerarquía.

Un esquema jerárquico de tipos de registros y tipos de IPH debe tener las siguientes propiedades:

1. Todo tipo de registros excepto la raíz participa como tipo de registros hijo en un y sólo un tipo de IPH.
2. Un tipo de registros puede participar como registro padre en cualquier número (cero o más) de tipos de IPH.
3. Si un tipo de registros participa como padre en más de un tipo de IPH, sus

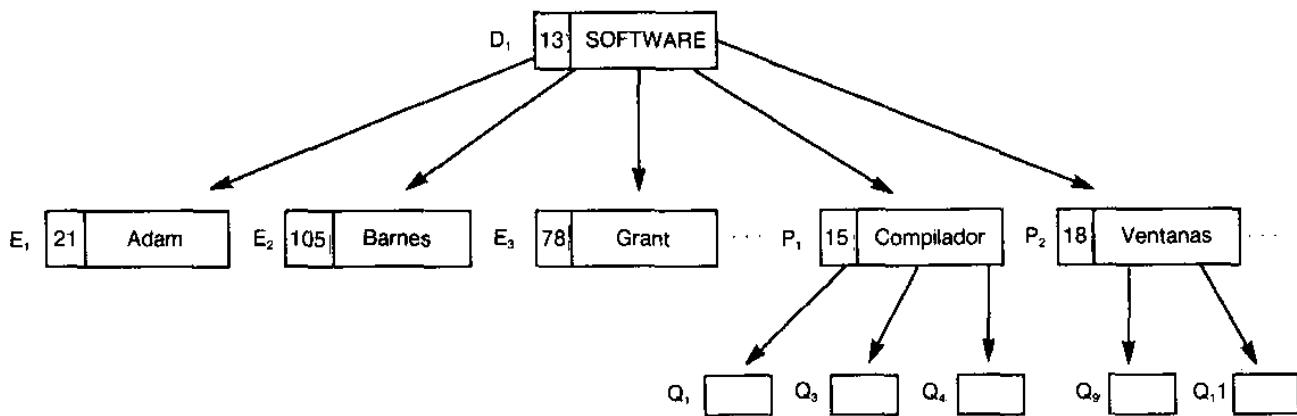


Figura 14.2. Un árbol de ocurrencia jerárquica.

tipos de registros hijos están ordenados. El orden se muestra, por convenio, de izquierda a derecha en un **diagrama de esquema jerárquico**.

Estas propiedades de un esquema jerárquico significan que cada tipo de registros excepto la raíz tiene exactamente un tipo de registros padre. Sin embargo, un tipo de registros puede tener varios tipos de registros hijos, que se ordenan de izquierda a derecha. En la figura 14.1 EMPLEADO es el primer hijo de DEPARTAMENTO, y PROYECTO es el segundo hijo. Estas propiedades limitan el tipo de interrelaciones que pueden estar representadas en un esquema jerárquico. En particular, las interrelaciones de muchos a muchos entre los tipos de registros *no se pueden* representar directamente, porque las interrelaciones padre-hijo son interrelaciones de uno a muchos y un tipo de registros no puede participar como hijo en dos o más interrelaciones distintas padre-hijo. Estas limitaciones causan problemas cuando se intenta definir el esquema jerárquico de una base de datos que contiene dichas interrelaciones no jerárquicas.

14.1.2. Arboles de ocurrencia jerárquicos y su almacenamiento lineal

En general, habrá muchas ocurrencias jerárquicas en la base de datos que correspondan al esquema jerárquico. Cada una de esas ocurrencias, también denominada *árbol de ocurrencia*, es una estructura de árbol cuya raíz es un solo registro del tipo de registros de raíz. El árbol de ocurrencia contiene también todas las ocurrencias de los registros hijo del registro raíz, todas las ocurrencias de los registros hijo en las IPH de cada uno de los registros hijo del registro raíz, y así sucesivamente hasta llegar a registros del tipo de registros hoja. Un árbol de ocurrencia del esquema en la figura 14.1 se muestra en la figura 14.2. Por conveniencia, se han etiquetado las ocurrencias de los tipos de registro E₁, E₂ y E₃ para empleados: P₁ y P₂ para proyectos, etcétera. En una base de datos real habría un árbol de ocurrencia por departamento.

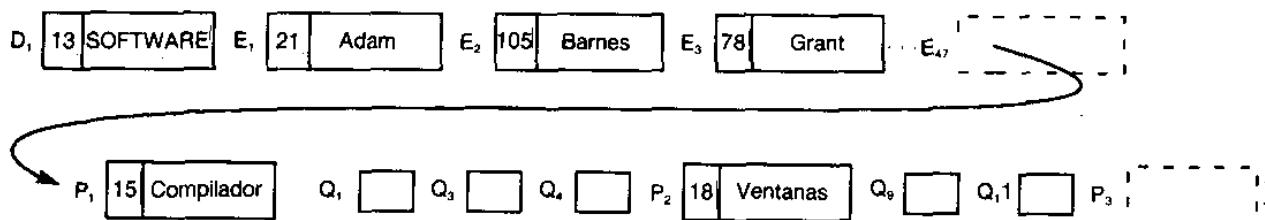


Figura 14.3. Almacenamiento lineal del árbol de ocurrencia.

La base de datos almacenada contiene estos registros en algún orden lineal, que corresponde a un *recorrido en preorden* del árbol de ocurrencia. Este recorrido puede definirse recursivamente afirmando que para recorrer un árbol con una raíz dada en un recorrido preorden, sus hijos deben ser visitados de izquierda a derecha, y los subárboles enraizados en estos hijos deben ser recorridos en preorden. En términos simples, un recorrido preorden corresponde a visitar el hijo antes que el *gemelo* o *hermano*, es decir, el siguiente nodo en el mismo nivel. El orden lineal del almacenamiento de los registros de la figura 14.2 se muestra en la figura 14.3. Los DBMS reales almacenan esta secuencia en bloques más que en registros individuales y permiten una variedad de opciones de indexación o indexación.

Una población entera de registros dentro de la base de datos jerárquica consta de una secuencia de los árboles de ocurrencia. Para una consulta determinada, la búsqueda en una base de datos jerárquica comúnmente localiza uno o más árboles de ocurrencia calificados y procesa árboles individuales con base en su almacenamiento lineal. Los DBMS jerárquicos también suelen incluir métodos para tener acceso a registros en niveles más bajos del esquema jerárquico, sea directamente o a través de índices.

14.1.3. Terminología de IMS

Debido a la popularidad del sistema jerárquico IMS, el modelo jerárquico se describe comúnmente en términos de IMS. Los tipos de registros se denominan *segmentos* en IMS. Una jerarquía pura de tipos de registros como se ha descrito hasta ahora se denomina *base de datos física*. Las interrelaciones padre-hijo se denominan *interrelaciones físicas padre-hijo*. Un árbol de ocurrencia se denomina *registro físico*. Posteriormente se introducirán más términos.

14.1.4. Tratamiento de interrelaciones de muchos a muchos

Las interrelaciones de muchos a muchos pueden gestionarse en el modelo jerárquico permitiendo la duplicación de casos de registros. Por ejemplo, considérese una interrelación de muchos a muchos entre empleado y departamento, donde un proyecto puede tener varios empleados trabajando en él, y un em-

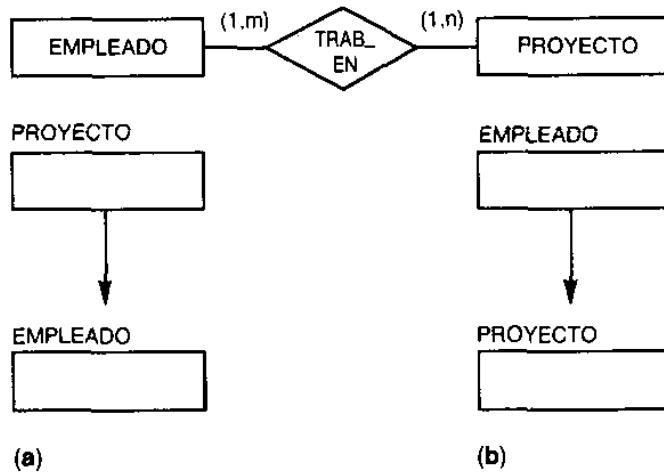


Figura 14.4. Representación de una interrelación de muchos a muchos en el modelo jerárquico.

pleado puede trabajar en varios proyectos. Se puede representar la interrelación como un tipo de IPH (departamento, proyecto), como se muestra en la figura 14.4a. En este caso, un registro que describa al mismo empleado se puede duplicar colocándolo una vez bajo cada proyecto en el que trabaje el empleado. Como alternativa, se puede representar la interrelación con un tipo de IPH (empleado, departamento), como se muestra en la figura 14.4b, en cuyo caso los registros de proyectos pueden ser duplicados.

La duplicación de registros, además de desperdiciar espacio de almacenamiento, crea el problema de mantener la congruencia (consistencia) entre las copias duplicadas. El concepto de tipo de registros *virtual* (*o puntero*) se utiliza en el sistema IMS para evitar las copias duplicadas de los registros. Esta técnica requiere la definición de interrelaciones *entre* esquemas jerárquicos, denominadas *interrelaciones virtuales padre-hijo*. Cuando se utilizan interrelaciones virtuales padre-hijo, el esquema resultante no sigue siendo estrictamente una jerarquía. Se explican estos conceptos en el próximo apartado.

14.1.5. Registros virtuales e interrelaciones virtuales padre-hijo

En esta sección se desarrollan los conceptos de manera más general, en lugar de limitarlos a la terminología IMS. Posteriormente, se comenta la manera en que IMS trata estos conceptos.

Un tipo de registros *virtual* (*o puntero*) HV es un tipo de registros con la propiedad de que cada uno de sus registros contiene un puntero a un registro perteneciente a otro tipo de registros, PV. HV desempeña la función de hijo virtual y PV la función de padre virtual en un tipo de *interrelaciones virtuales padre-hijo* (IVPH). Cada ocurrencia de registro H de HV apunta a exactamente una ocu-

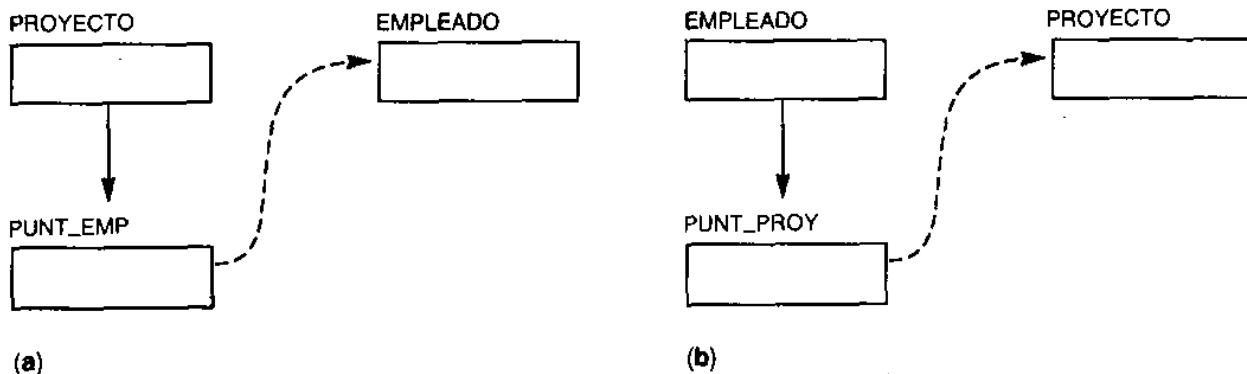


Figura 14.5. Registros virtuales e interrelaciones virtuales padre-hijo.

rrencia de registro P de PV . En lugar de duplicar el propio registro P en un árbol de ocurrencia, se incluye el registro virtual H que apunta a P . Es posible almacenar varios registros virtuales, cada uno de los cuales apunta a P , pero sólo una copia de P se almacena en la base de datos.

La figura 14.5 muestra dos maneras de representar la interrelación de muchos a muchos entre EMPLEADO y PROYECTO usando registros virtuales. La primera opción muestra el duplicado de EMPLEADO representado por PUNT_EMP, que tiene una interrelación virtual padre-hijo con EMPLEADO como padre virtual de un esquema jerárquico *diferente*. Compárese esto con la figura 14.4, donde se representó la misma interrelación sin registros virtuales. Supóngase que se tiene la siguiente situación de muestra:

PROYECTO EMPLEADOS

P_1	E_1, E_2, E_4
P_2	E_2, E_5
P_3	E_1, E_3

Según el esquema de la figura 14.4a, habrá duplicados de los empleados E_1 y E_2 , que trabajan en múltiples proyectos. Esta situación se evita en la figura 14.5a, donde el registro PUNT_EMP es un tipo de registros virtuales y su padre virtual es el registro EMPLEADO. La interrelación entre PUNT_EMP y EMPLEADO es entre *dos jerarquías independientes*. El esquema de la base de datos de la figura 14.5a contiene solamente una copia de cada registro EMPLEADO; sin embargo, varias ocurrencias de registros virtuales PUNT_EMP pueden apuntar al mismo registro empleado. Así, habrá dos ocurrencias de PUNT_EMP bajo los proyectos P_1 y P_3 , ambos apuntando al mismo registro empleado, a saber, el empleado E_1 . La información que depende de los dos registros, padre e hijo, como las horas que trabaja un empleado por semana en un proyecto, está incluida en el registro virtual puntero; tales datos se conocen comúnmente entre los usuarios de bases de datos jerárquicas como **datos de intersección**. Por tanto, las dos ocurrencias de PUNT_EMP que corresponden al empleado E_1 pueden contener datos adicionales.

nales sobre las horas de trabajo semanales del empleado E_i en los proyectos P₁ y P₃, respectivamente.

El esquema de la figura 14.4b se representa de manera similar sin duplicación en la figura 14.5b usando el tipo de registros virtual PUNT_PROY y definiendo PROYECTO como su padre virtual a través de una interrelación virtual padre-hijo. También existe una tercera opción, denominada *interrelaciones bidireccionales*, en la cual se pueden definir simultáneamente las dos interrelaciones virtuales padre-hijo para permitir un acceso eficiente de PROYECTO a EMPLEADO y de EMPLEADO a PROYECTO a través de los dos tipos de registros virtuales pero almacenando solamente un tipo de registros puntero que apunta en las dos direcciones.

14.1.6. Implantación de las interrelaciones virtuales padre-hijo

Conceptualmente, los tipos de IPH y los de IVPH son similares; la diferencia principal estriba en la manera de llevarlos a la práctica. Los tipos de IPH casi siempre se implantan utilizando la secuencia jerárquica de ocurrencia de padre seguida por las ocurrencias de tipos de registros hijo, mientras que los tipos IVPH se implantan usualmente teniendo un puntero de un registro hijo virtual a su registro padre virtual denominado **puntero a padre virtual**. Esto afecta principalmente la eficiencia de ciertas consultas.

Los IVPH se pueden implantar de diferentes maneras. Una opción es tener sólo un puntero en el hijo virtual que apunte al padre virtual, como se expuso anteriormente. Una segunda opción es tener, además del puntero de hijo a padre, un vínculo hacia atrás del padre virtual a una lista enlazada de registros hijo virtuales. El puntero del padre virtual al primer registro hijo virtual se denomina **puntero a hijo virtual**, mientras que un puntero de un hijo virtual al siguiente se denomina **puntero a gemelo virtual**. Dependiendo de los tipos de punteros disponibles, las interrelaciones virtuales pueden ser recorridas en cualquier dirección, como sucede con los tipos de conjuntos en el modelo de redes, que se expuso en el capítulo 13. Este vínculo hacia atrás facilita la recuperación de todos los registros hijo virtuales de un cierto registro padre virtual.

En general, hay muchos métodos posibles de diseñar una base de datos usando el modelo jerárquico. En muchos casos las consideraciones de rendimiento son los factores más importantes para seleccionar un esquema jerárquico de base de datos en lugar de otro. La ejecución depende de las opciones de implantación, como punteros o índices, que estén disponibles en cada sistema específico, así como de los métodos de agrupamiento jerárquico.

14.1.7. Más terminología IMS

Los tipos de registros se denominan *segmentos* en IMS. IMS llama a los registros virtuales *segmentos punteros* y a las interrelaciones virtuales padre-hijo *inter-*

relaciones padre-hijo lógicas. IMS denomina al hijo de una interrelación padre-hijo real *hijo físico*, y al hijo de una interrelación padre-hijo virtual, *hijo lógico*. IMS requiere que cada aplicación defina la porción pertinente de una o más bases de datos jerárquicas como subjerarquía o base de datos lógica (véase el apartado 14.2.5) y compilarla como *bloque de comunicación de programa* (*PCB, program communication block*).

14.2. Correspondencia de esquemas del modelo ER al modelo jerárquico

La correspondencia del modelo ER al jerárquico no es tan directa y sistemática como la correspondencia al modelo relacional o al de redes. Esto es debido a que un esquema ER es esencialmente una gráfica con nodos que son los tipos de entidades y vínculos representados por los tipos de interrelaciones. Para convertir la gráfica en un esquema relacional, la «aplanamos» rompiendo todos los vínculos y representando la información de los nodos y los vínculos rotos en forma de tablas. Para convertir la gráfica en un esquema de redes, se mantuvo la estructura de gráfica (véase el apartado 13.2); la información de los nodos se representó con tipos de registros, y la mayoría de los vínculos se transformaron en tipos de conjuntos. Hay, por supuesto, algunos casos especiales de interrelaciones de muchos a muchos que se resuelven a través de registros vínculo, etcétera.

La transformación de un esquema ER en uno jerárquico equivale a hacer corresponder la estructura de gráfica con una o más estructuras tipo árbol. En este proceso influye la semántica de la aplicación, sobre todo la selección de nodos raíz, las decisiones sobre qué interrelaciones modelar como interrelaciones normales padre-hijo y cuáles como interrelaciones virtuales padre-hijo, y así sucesivamente. Primero trataremos la transformación suponiendo que estamos generando *jerarquías puras*; esto es, haremos que los registros se dupliquen de manera que ninguno tenga múltiples padres; a continuación, conectaremos las jerarquías usando las interrelaciones virtuales padre-hijo. Primero presentaremos un procedimiento generalizado y luego transformaremos el esquema ER ejemplo de la figura 13.7. Las construcciones específicas (por ejemplo, los atributos polivalentes) que no sean tratadas en el procedimiento general se tratarán más adelante.

La generación de un esquema jerárquico óptimo es un problema difícil que ha sido tema de investigaciones (véanse las referencias a los trabajos de Sakai y de Navathe y Cheng en la bibliografía). El procedimiento que presentamos en seguida genera un esquema posible más que uno óptimo. Para lograr correspondencias óptimas es necesario un procedimiento más complicado.

14.2.1. Procedimiento general para la transformación de esquemas ER a jerárquicos

Este procedimiento genera primero una estructura tipo redes a partir del esquema ER, y luego crea tipos de registros duplicados para generar jerarquías; por último, se utilizan interrelaciones padre-hijo virtuales para conectar esas jerarquías. No nos vamos a referir a ningún modelo jerárquico específico, pero el procedimiento genera esquemas que se pueden poner en práctica en cualquier DBMS jerárquico, incluidos IMS o System 2000.

Paso 1. Crear un tipo de registros para cada tipo de entidades del esquema ER. Los atributos se convierten en elementos (o *campos* en IMS) del registro.

Paso 2. Para cada interrelación de uno a muchos de E_1 a E_2 , con los registros correspondientes J_1 y J_2 , crear una interrelación padre-hijo de J_1 a J_2 . No se espera que la interrelación tenga atributos; si existe alguno, se debe representar en J_2 .

Paso 3. Para cada interrelación R de uno a uno entre E_1 y E_2 , con los registros correspondientes J_1 y J_2 , elegir una de estas dos opciones:

- a) Crear una interrelación padre-hijo en cualquier dirección entre J_1 y J_2 ; transferir los atributos de R bien a J_1 o a J_2 , o a ambos (con la carga adicional de mantenimiento de la coherencia). El tipo de registros al que normalmente se obtenga acceso primero en las aplicaciones sería escogido como padre.
- b) Combinar los dos tipos de registros J_1 y J_2 en uno y representar cualesquiera atributos de R en el registro resultante.

Paso 4. Para cada interrelación R de muchos a muchos entre E_1 y E_2 , con los registros correspondientes J_1 y J_2 , crear un tipo de registros adicional J_3 y representar cualesquiera atributos de R en J_3 . En seguida, crear dos interrelaciones padre-hijo: una de J_1 a J_3 y la otra de J_2 a J_3 .

Paso 5. Para cada interrelación n-aria R entre la entidades E_1, E_2, \dots, E_n , crear un tipo de registros para representar la interrelación, y representar cualesquiera atributos de R en este registro. En seguida, definir n interrelaciones padre-hijo de los correspondientes n tipos de registro como padres, con este tipo de registros como hijo de cada uno.

Llegados a este punto hemos transformado el esquema ER original en una estructura tipo redes con varios registros que pueden tener múltiples padres.

Paso 6. Este paso se ocupa de los tipos de registros del esquema actual que tienen más de una interrelación padre-hijo apuntando a cada uno de ellos. Primero se determina si la *heurística de inversión de la jerarquía* definida a conti-

nuación se aplica. Siempre que sea posible, se aplicará y se invertirá la (sub)jerarquía, usando el registro vínculo como raíz.

Heurística de inversión de la jerarquía: Si un tipo de registros R (registro vínculo) tiene dos (o más) interrelaciones que apuntan a él desde tipos de registros S y T (y otros), y S y T (y otros), a su vez, no tienen interrelaciones apuntando hacia ellos, se puede tratar R como un tipo de registros padre, y S y T (y otros) como sus hijos. Si la jerarquía resultante con el registro vínculo como raíz tiene sentido y es razonable, decimos que es aplicable la heurística de inversión de la jerarquía.

Paso 6a. Este paso se aplica a los DBMS jerárquicos que permiten interrelaciones virtuales padre-hijo (IMS las denomina *interrelaciones lógicas*). Obsérvese que en IMS sólo se permite un padre virtual y uno real para cada tipo de registros hijo. Por tanto, los casos de dos interrelaciones apuntando al mismo registro y los de más de dos deben ser tratados aparte.

Caso 1: Dos interrelaciones inciden en el mismo registro (lo llamaremos *registro vínculo*). Se define un tipo de registros que corresponde al registro vínculo y se coloca debajo del padre que tiene la relación más estrecha, denominado **padre primario**, de los dos posibles padres, visto desde la perspectiva de la aplicación. Para esta determinación puede ayudarnos una estimación cuantitativa del número medio de accesos requeridos a los dos padres, según las pautas de la explicación del capítulo 11. El padre a través del cual se obtiene acceso al registro vínculo con más frecuencia se declara el padre primario. El registro vínculo también sirve como hijo virtual que apunta al otro padre como padre virtual en una interrelación virtual padre-hijo.

Caso 2: Más de dos interrelaciones (digamos, m interrelaciones) inciden en el mismo registro vínculo. Se define un tipo de registros que corresponda al registro vínculo y se coloca debajo del padre primario, que se selecciona como se describió en el caso 1. Como sólo se permite un padre virtual, se necesita crear $m - 1$ **copias secundarias** de este registro vínculo y colocarlas como hijos debajo del registro vínculo en el esquema con el único propósito de facilitar el enlace con los $m - 1$ padres virtuales. No es preciso que las copias secundarias contengan algún elemento; opcionalmente, se puede duplicar en ellas cualquier elemento del registro vínculo (con la carga añadida de mantenimiento de la coherencia). Cada copia secundaria apunta entonces al respectivo tipo de registros padre como padre virtual en una interrelación virtual padre-hijo. Así se captan las m interrelaciones y todavía se cumple la restricción de que un registro vínculo puede tener como máximo dos padres, uno real (en IMS, *físico*) y otro virtual (en IMS, *lógico*).

Paso 6b. Este paso se aplica a los DBMS jerárquicos que no permiten interrelaciones virtuales padre-hijo (por ejemplo, System 2000). Para cada tipo de registros del esquema actual que tenga m interrelaciones padre-hijo apuntando a él ($m > 1$), crear m copias duplicadas. Volver a definir las interrelaciones ori-

ginales padre-hijo de manera que cada copia tenga ahora sólo una interrelación padre-hijo apuntando a ella desde un padre apropiado.

Dependiendo del esquema específico en cuestión, es posible emplear una combinación de las estrategias anteriores, mientras las restricciones de modelo sean obedecidas y el significado original del esquema conceptual sea preservado.

Paso 7. El esquema resultante podría contener ahora una o más jerarquías interconectadas por interrelaciones virtuales padre-hijo. Si se desea un solo esquema jerárquico, se puede crear un registro raíz ficticio que se convierta en el padre de todos los nodos de raíz de las jerarquías en cuestión.

14.2.2. Ejemplo: La base de datos de control de pedidos

Se aplicará el procedimiento anterior al esquema ER que se muestra en la figura 13.7. En el paso 1 todas las entidades se transforman en tipos de registros. En el paso 2 se tratan las interrelaciones de uno a muchos: (REGION, SUCURSAL), (SUCURSAL, CLIENTE), (SUCURSAL, VENDEDOR) y (VENDEDOR, PEDIDO). Se convierte cada una en una interrelación padre-hijo. En el paso 3 la interrelación de uno a uno (CLIENTE, INFO_ENVIO) se transforma en una interrelación padre-hijo con cliente como padre, ya que normalmente se obtiene acceso a él antes que a INFO-ENVIO. El esquema jerárquico resultante en esta etapa se muestra en la figura 14.6.

En este punto el esquema parece jerárquico, aunque no esté completo porque las interrelaciones de muchos a muchos y n-arias todavía no han sido representadas. El paso 4 trata las interrelaciones de muchos a muchos entre SUCURSAL y PRODUCTO, y entre PEDIDO y PRODUCTO. Se crean dos registros nuevos, SUCURSAL_PRODUCTO y PEDIDO_PRODUCTO respectivamente, para estas dos interrelaciones, seguidas cada una por un par de interrelaciones padre-hijo. Finalmente, correspondiendo a la interrelación ternaria CVP, se crea un tipo de registros adicional, TRANSACCION_PEDIDO, y tres nuevas interrelaciones padre-hijo. La figura 14.7 muestra el esquema resultante al final de la etapa 5; parece un esquema de redes.

En el paso 6 se tratan los tipos de registros que tienen múltiples padres en la figura 14.7: SUCURSAL_PRODUCTO, PEDIDO_PRODUCTO y TRANSACCION_PEDIDO. Supongamos que se está haciendo un diseño lógico para un sistema tipo IMS que permite interrelaciones padre-hijo virtuales.

Los tipos de registro SUCURSAL_PRODUCTO y PEDIDO_PRODUCTO son registros vínculos para los cuales se debe considerar si se aplica la heurística de inversión jerárquica. Si se invierten estas subjerarquías, los tipos de registros SUCURSAL_PRODUCTO y PEDIDO_PRODUCTO se convertirán en raíces de las subjerarquías respectivas, que actuarán como bases de datos independientes. No se anticipa la necesidad de obtener acceso a la base de datos de control de pedidos con estos registros como puntos primarios de entrada para iniciar una

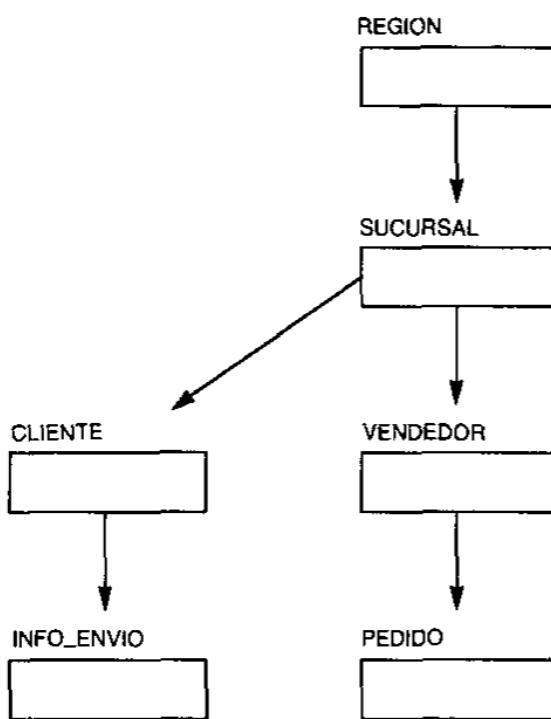


Figura 14.6. Esquema resultante producido a partir del esquema ER de la figura 13.7. después del paso 3 de correspondencia jerárquica.

búsqueda. Por tanto, se considera poco práctica esta opción y se rechaza la aplicación de la heurística.

Se aplica la correspondencia del paso 6a. Los tipos de registros SUCURSAL_PRODUCTO y PEDIDO_PRODUCTO tienen cada uno dos interrelaciones padre-hijo incidiendo en ellos. Por el procedimiento del caso 1, se selecciona PRODUCTO como padre primario del registro vínculo SUCURSAL_PRODUCTO porque se espera que sea más probable requerir acceso a las sucursales donde se vende un producto que a todos los productos que vende una sucursal. Por lo mismo, se escoge PEDIDO como padre primario del registro vínculo PEDIDO_PRODUCTO. Establecemos interrelaciones virtuales padre-hijo para que SUCURSAL_PRODUCTO apunte a su padre virtual, SUCURSAL, y que PEDIDO_PRODUCTO apunte a su padre virtual, PRODUCTO.

En el caso del tipo de registros TRANSACCION_PEDIDO se tienen tres padres: CLIENTE, VENDEDOR (el que recibe la comisión, que puede ser diferente del que hizo el pedido), y PEDIDO; se escoge PEDIDO como padre primario de este tipo de registros. Se crean dos copias secundarias denominadas TP₁ y TP₂, para representar el tipo de registros TRANSACCION_PEDIDO, colocándolas como hijos de TRANSACCION_PEDIDO. Una de ellas apunta a su padre virtual, cliente, y la otra a su padre virtual, VENDEDOR¹. Los resultados del paso 6 son las dos jerarquías

¹ El padre virtual, VENDEDOR, ???... considerando la relación ternaria original CVP. En la vida real, esto implica que el vendedor que hizo el pedido y el que realiza la transacción del pedido pueden ser individuos diferentes.

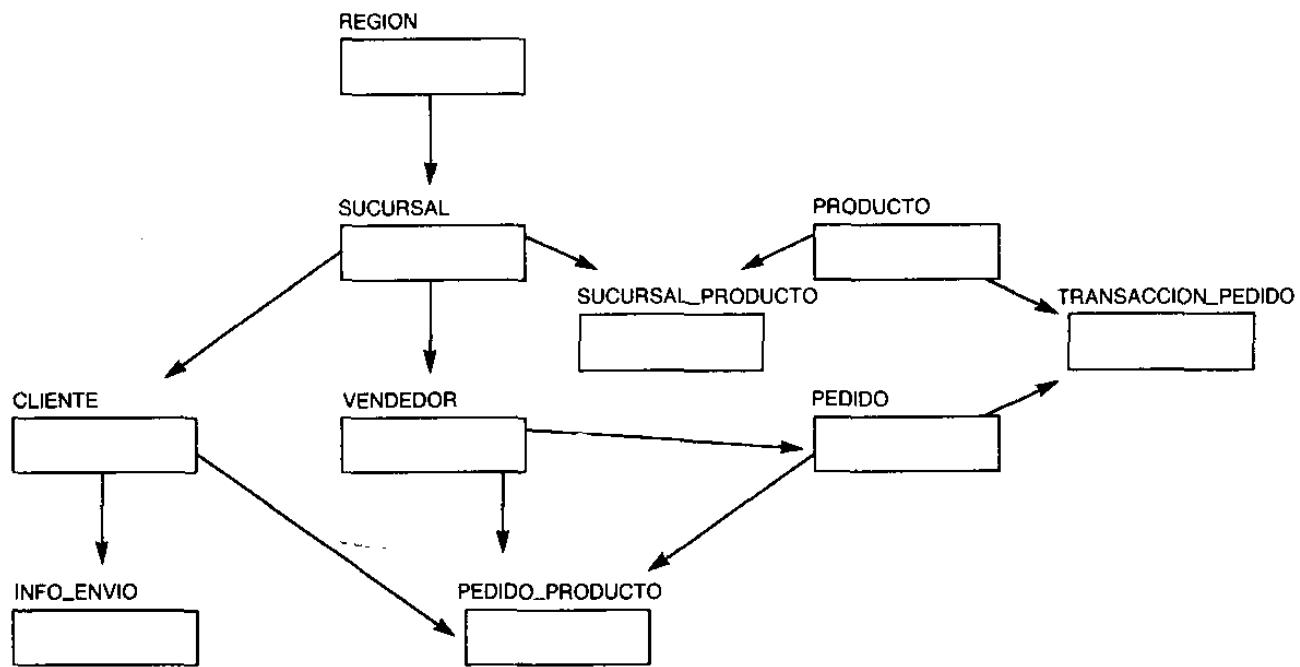


Figura 14.7. Esquema resultante después del paso 5 de la correspondencia jerárquica.

que se muestran en la figura 14.8, con los tipos de registro raíz REGION y PRODUCTO. En IMS estas dos jerarquías serían definidas como dos *bases de datos físicas* (*PDB, physical databases*). Si se quiere, puede colocarse un registro ficticio sobre los dos registros raíz para producir una sola base de datos jerárquica, que se convierte en una sola base de datos física en IMS. Este registro ficticio se rotula apropiadamente y se representa con un rectángulo de guiones en la figura 14.8.

14.2.3. Transformación de situaciones especiales

En los apartados 12.2 y 13.2 se enfocó el problema de correspondencia de un esquema conceptual ER a esquemas en los modelos relacional y de redes como un problema de resolver individualmente la correspondencia de ciertas situaciones especiales. Aquí, nuestro enfoque fue diferente: primero dimos un procedimiento general para hacer corresponder las entidades con tipos de registros y las interrelaciones con interrelaciones padre-hijo reales y virtuales. Ahora se considerarán una a una las situaciones de modelado restantes.

Atributos compuestos. No hay un procedimiento especial para tratar los atributos compuestos en el modelo jerárquico, como lo hay en el modelo de redes. En IMS podemos asignar al atributo compuesto un nombre de campo, y

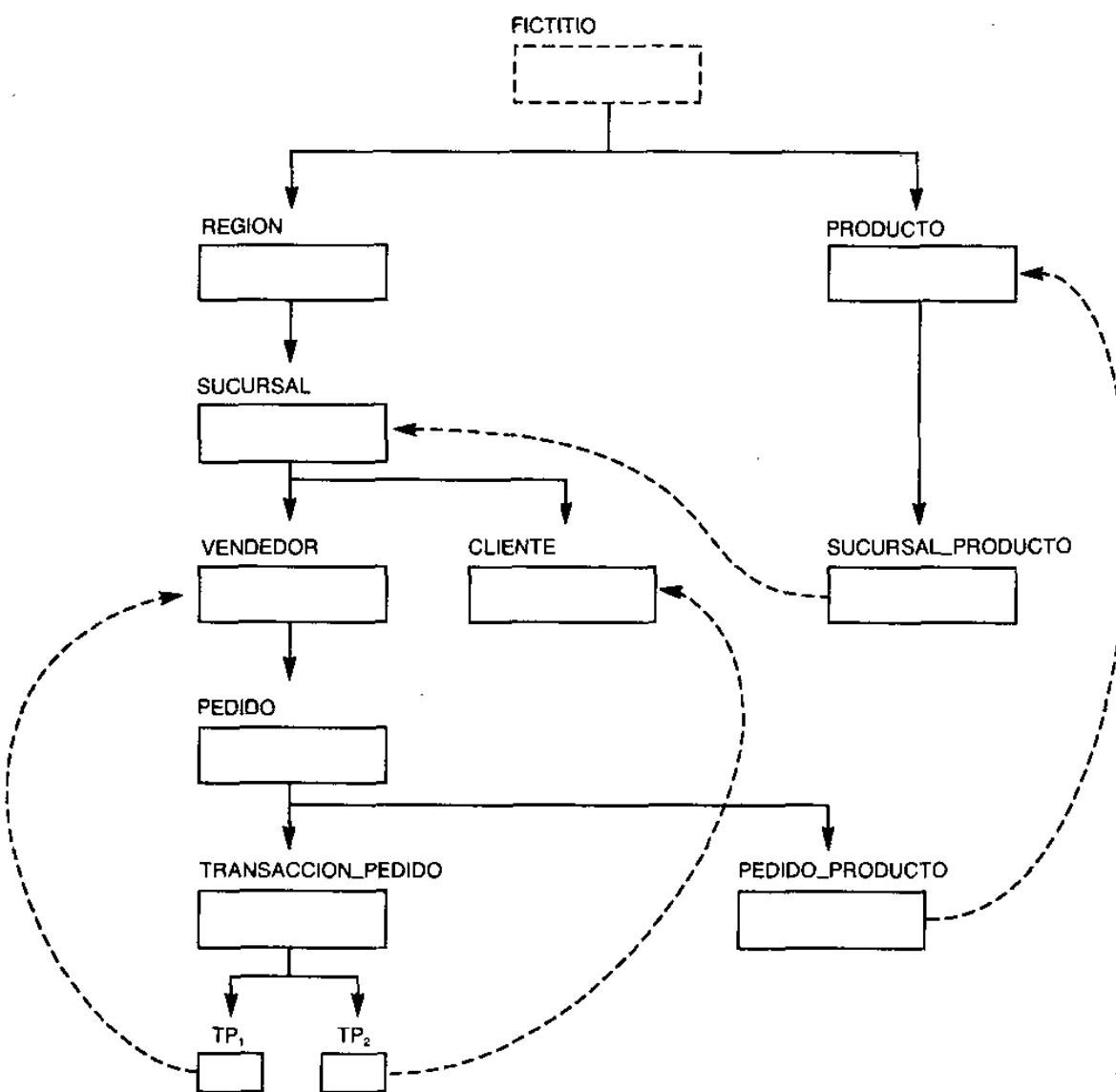
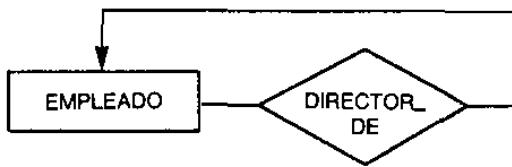


Figura 14.8. Resultado final de la correspondencia jerárquica.

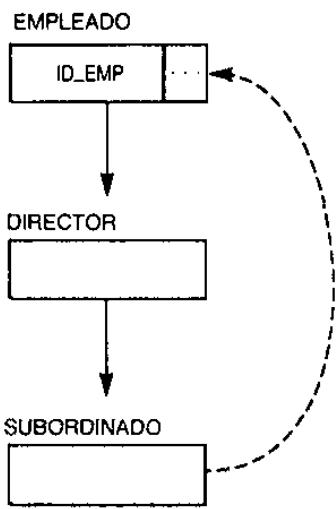
definir sus atributos componentes dentro de él como campos individuales. En System 2000 se define un **grupo de repetición**, como en el modelo de redes, que puede tener una o más ocurrencias.

Atributos polivalentes. No están contemplados los elementos de datos tipo vector en el modelo jerárquico. Los grupos de atributos que se repiten se tratan exactamente como en los sistemas de redes: se crea un grupo de repetición dentro de un tipo de registros, o bien se crea un tipo de registros nuevo y se relaciona con el padre a través de una interrelación padre-hijo.

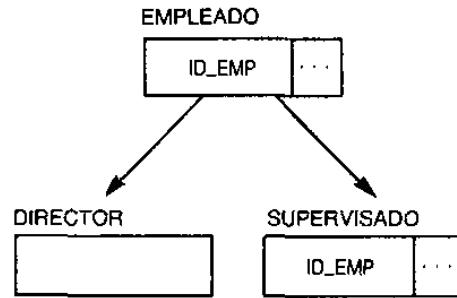
Identificadores externos. La identificación externa del hijo desde el padre



Interrelación recursiva en el modelo ER



(a) Opción 1



(b) Opción 2

Figura 14.9. Transformación de interrelaciones recursivas en el modelo jerárquico.

está supuesta implícitamente en el modelo jerárquico. Por tanto, en el esquema de la figura 14.8, si SUCURSAL tiene un identificador externo CODIGO_REGION procedente de REGION, *no hay necesidad* de incluirlo en SUCURSAL. Esta es una ventaja especial del modelo jerárquico: la identificación externa se propaga hacia abajo por la jerarquía. Así, todo registro tiene implícitamente una clave, denominada *clave jerárquica*, que es una concatenación de las claves de todos los antepasados jerárquicos combinada con la clave propia del registro.

Interrelaciones recursivas. Considérese la interrelación recursiva, o de anillo, denominada DIRECTOR_DE entre la entidad EMPLEADO y ella misma (Fig. 14.9). En el modelo jerárquico se representa creando un tipo de registros director y una interrelación padre-hijo de EMPLEADO a DIRECTOR. Sólo aquellos empleados que sean directores tendrán una ocurrencia hijo de DIRECTOR bajo ellos. Si se desea encontrar los empleados que un director supervisa, se puede crear un registro SUBORDINADO que es hijo de DIRECTOR y tiene una interrelación padre-hijo virtual con EMPLEADO en la cual es el padre. Esta situación se muestra en la figura 14.9a. Otra posibilidad está modelada en la figura 14.9b, donde hay dos registros hijo bajo EMPLEADO: uno igual que el DIRECTOR de antes y uno

para SUPERVISADO. Ambos existen sólo si un empleado es director; en cuyo caso el primero tiene una ocurrencia, y el segundo tiene muchas.

14.2.4. Algunas restricciones de diseño lógico para IMS

El esquema resultante después de la transformación precedente está sujeto a las restricciones del DBMS objetivo. Como IMS es todavía el DBMS jerárquico dominante, se mencionan varias de las reglas aplicables.

- IMS restringe el número de segmentos (comúnmente a 225) y el número de niveles de la jerarquía (comúnmente a 15) en una base de datos jerárquica.
- Un tipo de registros puede tener un padre físico (a través de una interrelación padre-hijo) y un padre lógico (a través de una interrelación virtual padre-hijo).
- Un segmento de hijo lógico no puede tener un hijo lógico.
- Un segmento raíz de una jerarquía (denominada una *base de datos física* en IMS) no puede ser un hijo lógico.

Debido a las restricciones anteriores, las siguientes consideraciones se aplican *después* de la correspondencia:

- Si un registro padre tiene un hijo en el esquema, y el tipo de registros padre tiene muchos menos datos que el tipo de registros hijo, o a la inversa, existe la posibilidad potencial de integrar el registro padre en el hijo, creando un solo registro.
- Siempre que hay una interrelación lógica, existe la posibilidad de eliminar el hijo lógico completamente porque representa una copia adicional (aunque con muchos menos datos, o incluso ninguno) y constituye un gasto adicional. El costo de la eliminación es la pérdida de una interrelación que podría ser captada por otras alternativas, como almacenar claves de registros relacionadas dentro de un registro dado.

14.2.5. Bases de datos lógicas en IMS

Es posible seleccionar de un esquema jerárquico dado sólo los tipos de registros pertinentes a una aplicación y definir una **subjerarquía**, o una vista de una jerarquía. Una vez definidos varios esquemas jerárquicos y las interrelaciones virtuales padre-hijo entre ellos, es posible generar varias jerarquías diferentes combinando algunos de sus registros e incluyendo tanto IPH como IVPH. Estas pueden denominarse **vistas de esquemas jerárquicos**. Las diferentes aplicaciones pueden definir sus propias vistas.

En IMS, cada vista de una o más jerarquías se define y se nombra como una

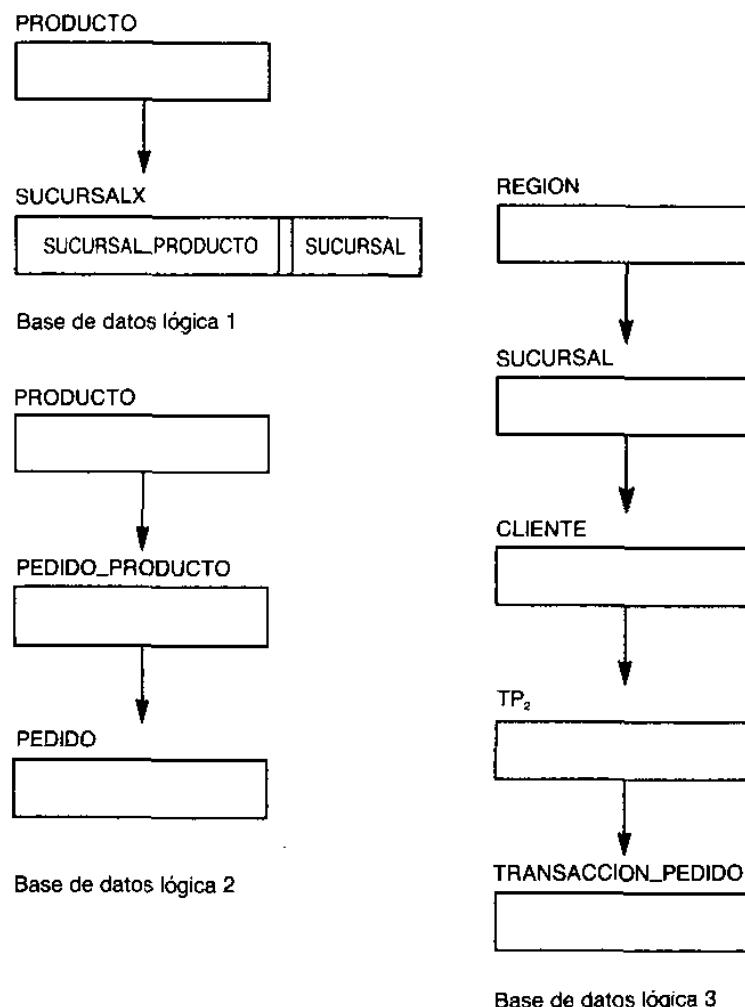


Figura 14.10. Ejemplo de bases de datos lógicas definidas sobre la base de datos de control de pedidos.

base de datos lógica. Aquí no profundizará en la mecánica de definir bases de datos lógicas en IMS. Sin embargo, la figura 14.10 ilustra cómo se puede definir tres vistas diferentes o bases de datos lógicas sobre el esquema jerárquico de la base de datos de control de pedidos de la figura 14.8. Obsérvese que al formar una nueva base de datos lógica, los segmentos (registros) que participan en una interrelación padre-hijo virtual algunas veces se consideran como si estuvieran concatenados en un solo tipo de registros, y se representan juntos con algún nombre nuevo. Los detalles de la definición de este tipo de estructuras usando el lenguaje DL/1 de IMS están más allá de nuestro ámbito.

La concatenación se ilustra en el primer esquema lógico de la figura 14.10, que esencialmente coloca **SUCURSAL** como subordinado de **PRODUCTO** y permite encontrar, digamos, todos los productos vendidos por una sucursal. Se da

al tipo de registros concatenado el nombre SUCURSALX para distinguirlo del tipo de registros original SUCURSAL.

También es posible tratar al padre (físico) de un registro como si fuera un hijo al formar el esquema lógico. Esto se debe a que una ocurrencia de hijo siempre tiene una ocurrencia única de su tipo de registros padre y, por tanto, el padre es accesible para un hijo dado. Por ejemplo, en el segundo esquema lógico, PEDIDO está colocado como subordinado de PEDIDO_PRODUCTO, aunque es el padre de éste en el esquema original. Esto permitirá encontrar todos los pedidos relacionados con un producto dado. De manera similar, en el tercer esquema lógico se colocó TRANSACCION_PEDIDO subordinado a TP₂. Este esquema lógico permite procesar todas las transacciones de pedidos para un cliente dado a través de la *copia secundaria* TP₂.

Obsérvese que las interrelaciones virtuales padre-hijo pueden ser recorridas en cualquier dirección para formar el esquema lógico. Se mostrará la base de datos lógica apropiada cuando se transformen las muestras de esquemas de navegación de la base de datos del caso de estudio en el apartado 14.4.2.

14.3. Correspondencia de operaciones del modelo ER al modelo jerárquico

En este apartado se muestran algunos ejemplos de transformación de especificaciones de esquemas de navegación del modelo ER en las construcciones típicas del lenguaje jerárquico de manipulación de datos (DML). No ha habido una propuesta para un DML jerárquico similar a la del DML de redes que se hizo en 1971 en el informe del DBTG de CODASYL. Por tanto, otra vez utilizaremos el lenguaje DL/1 de IMS para propósitos ilustrativos. Los lectores interesados pueden consultar el libro de Elmasri y Navathe (1989, Cap. 10), donde se propone un lenguaje neutral denominado HDML para el modelo jerárquico general. Se explica el lenguaje DL/1 y se compara con el HDML en el capítulo 23 de ese libro (véase la bibliografía del capítulo 1).

La tabla 14.1 resume las operaciones de DML provistas en IMS². Su descripción es sencilla. Se debe entrar a una jerarquía por el tipo de registros raíz y procesarse utilizando una variación de la operación GET. El programa de aplicación tiene su propia área de trabajo del usuario (UWA, *User Work Area*), donde se preparan los registros para inserción y se reciben los registros recuperados. Como el DML de redes, el DML jerárquico es un lenguaje orientado a la navegación. La navegación está limitada a un recorrido de la jerarquía. Una recuperación implica moverse de registro en registro (usando variaciones de GET) a lo largo de los caminos jerárquicos hasta localizar el registro que se requiere. No hay distinción entre FIND y GET, como en el DML de redes; sólo se utiliza

² Sólo se han incluido las operaciones esenciales del lenguaje DL/1 de IMS. Las operaciones pueden ser modificadas con variaciones como la de códigos de otro mandato.

Tabla 14.1. Resumen de mandatos de DML jerárquicos

Operación	Función
RECUPERACION	
GET	Se usa para recuperar el registro actual y colocarlo en la correspondiente variable del área de trabajo del usuario
GET HOLD	Se usa para recuperar el registro de un tipo dado y retenerlo como registro actual de manera que pueda ser subsecuentemente borrado o sustituido
<i>Las variaciones de GET y GET HOLD en IMS son:</i>	
GET UNIQUE	Se usa para recuperar un registro con base en condiciones; también se usa para la iniciación o entrada a un esquema
GET NEXT	Se usa para recuperar el próximo registro de un tipo dado
GET UNIQUE/NEXT (con código de mandato D*)	Se usa para recuperar una trayectoria jerárquica de registros
NAVEGACION	
GET NEXT (un registro)	Se usa para recuperar el próximo registro de un tipo dado
GET NEXT (para una serie de registros)	Se usa para recuperar la próxima ocurrencia de una trayectoria de registros jerárquicos
GET NEXT WITHIN PARENT	Se usa para recuperar el hijo específico del actual registro padre
ACTUALIZACION DE REGISTROS	
INSERT	Almacena el registro nuevo en la base de datos, tomando del área de trabajo del usuario, y lo convierte en el registro actual
DELETE	Elimina de la base de datos el registro actual
REPLACE	Reemplaza el registro actual por un nuevo registro tomado del área de trabajo del usuario.

GET, que fija la actualidad además de transferir los datos (generalmente un registro o múltiples registros con el código de mandato D*).

La actualización de los registros utiliza una estrategia similar de navegación a través de los tipos de registros y de interrelaciones padre-hijo hasta que el registro o los registros por actualizar sean localizados. El control real de la navegación, verificación de los códigos de estado, etc., lo realiza el programa de aplicación. Se usa DB_STATUS como variable disponible para la comunicación entre el programa de aplicación y el DBMS, y se supone que un valor de cero para DB_STATUS indica una operación satisfactoria. La actualización de los registros se practica usando los comandos INSERT, DELETE, y REPLACE. Las operaciones del tipo de las agregaciones se ejecutan utilizando los recursos del lenguaje de programación del computador.

Consulta 1: Encontrar los instructores que tienen menos de 30 años de edad

```
% GET UNIQUE INSTRUCTOR
  WHERE EDAD < 30
mientras DB_STATUS=0 hacer
  comenzar
    escribir información del instructor
%GET NEXT INSTRUCTOR
  WHERE EDAD < 30
terminar;
```

Figura 14.11. Especificación en DML jerárquico de una recuperación condicional de un tipo de entidad (la especificación de navegación ER es igual a la de la figura 13.10).

El concepto de **actualidad** también se tiene en el modelo jerárquico. El sistema guarda un indicador de actualidad para cada tipo de registros y sabe el lugar actual donde una aplicación está situada. La opción **HOLD** de la operación **GET** permite al registro recuperado mantenerse actualizado antes de ser borrado o sustituido.

La figura 14.11 muestra la correspondencia de una especificación de navegación ER simple sobre la entidad **INSTRUCTOR**. Por simplificar, las operaciones del DML jerárquico se muestran como mandatos precedidos de un signo % y utilizando nuestra propia pseudosintaxis. En IMS, las operaciones se ejecutan haciendo llamadas al sistema desde el programa de aplicación con el código de operación, las variables del área de trabajo del usuario, las condiciones, etc., como parámetros. Se muestran como enunciados con pseudomandatos. La condición «menos de 30 años de edad» corresponde a una selección de **EDAD** con la cadena **EDAD < 30**, que *puede* pasarse directamente como parámetro a la operación **GET UNIQUE**.

En la figura 14.12 el esquema de navegación requiere acceso a **CURSO** con una navegación de **INSTRUCTOR** a curso a través de la interrelación **OFRECE**. El

Consulta 2: Dado el número de instructor (\$NSS), encontrar los departamentos en los cuales ofrece cursos

```
% GET UNIQUE INSTRUCTOR
  WHERE INSTRUCTOR.NSS=$NSS;
mientras DB_STATUS=0 hacer
  comenzar
    /* el padre actual es el registro instructor*/
    % GET NEXT WITHIN PARENT CURSO
    escribir CURSO.DEPARTAMENTO
terminar;
```

Figura 14.12. Especificación en DML jerárquico de una operación de recuperación de múltiples tipos de entidades (la especificación de navegación ER es igual a la de la figura 13.11).

modelo jerárquico trata OFRECE como una interrelación padre-hijo sin nombre. Esta operación se transforma en un GET UNIQUE para INSTRUCTOR usando el valor dado de NSS, seguido por una operación GET NEXT WITHIN PARENT (obtener el siguiente dentro del parent) para recuperar los registros hijo CURSO. Obsérvese que el proceso de iteración a través de todos los cursos que son hijos bajo la ocurrencia parent actual de INSTRUCTOR lo controla el lenguaje de programación anfitrión.

14.4. Base de datos del caso de estudio en el modelo jerárquico

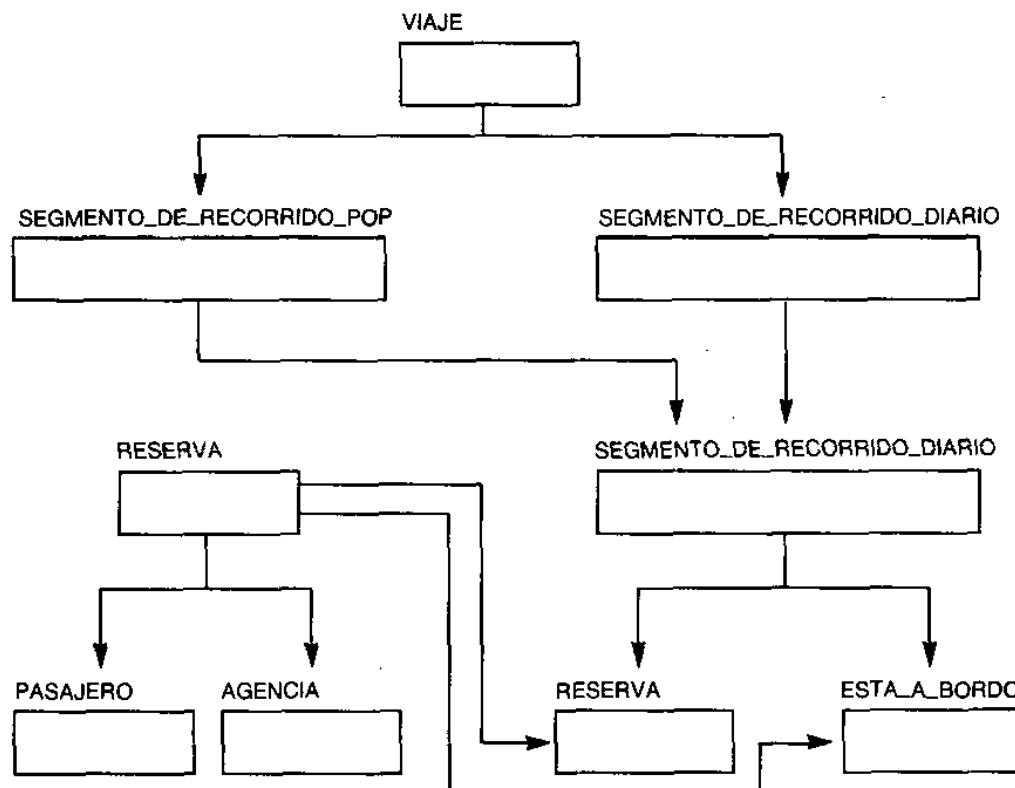
Continuamos ahora con el caso de estudio de capítulos anteriores. Se aplica la metodología mostrada en el apartado 14.2 para generar el esquema lógico jerárquico. Luego se muestra la transformación de algunos ejemplos de operaciones de navegación al DML jerárquico.

14.4.1. Correspondencia de esquemas con el modelo jerárquico

La correspondencia del esquema conceptual a lógico (datos para el caso de estudio) de la figura 11.15 con un esquema jerárquico se realiza de acuerdo con el procedimiento del apartado 14.2. Se empieza por hacer corresponder todas las entidades con tipos de registros y luego se convierten las interrelaciones de uno a uno y de uno a muchos en interrelaciones padre-hijo únicas. Esto tiene como resultado una jerarquía con CLIENTE como raíz y PASAJERO y AGENCIA como sus hijos. Otra jerarquía tiene VIAJE como raíz, los tipos de registros SEGMENTO_DE_RECORRIDO_POP y SEGMENTO_DE_RECORRIDO_ORD en el siguiente nivel, y más abajo las dos interrelaciones padre-hijo que inciden desde esos dos tipos de registros sobre un mismo tipo de registros, SEGMENTO_DE_RECORRIDO_DIARIO. En el paso 4 se crean los tipos de registros adicionales RESERVA y ESTÁ_A_BORDO para ocuparnos de las dos interrelaciones de muchos a muchos entre CLIENTE y SEGMENTO_DE_RECORRIDO_DIARIO. Con estas conversiones, el resultado intermedio al final del paso 5 (que tiene una estructura de redes) se muestra en la figura 14.13a.

La figura muestra cómo los registros SEGMENTO_DE_RECORRIDO_DIARIO, RESERVA y ESTÁ_A_BORDO tienen dos interrelaciones padre-hijo incidentes en ellos. En el paso 6 se aplica la heurística de inversión de la jerarquía, convirtiendo a RESERVA en una raíz de la jerarquía con CLIENTE y SEGMENTO_DE_RECORRIDO_DIARIO como sus hijos. Esto es sensato porque en el caso de estudio hay una necesidad frecuente de que las reservas sean fácilmente accesibles. De manera similar, ESTÁ_A_BORDO también se hace una raíz de la jerarquía con CLIENTE y SEGMENTO_DE_RECORRIDO_DIARIO como hijos. El resultado de esta inversión de la jerarquía se muestra en la figura 14.13b.

Finalmente, en el paso 6 se centra la atención en SEGMENTO_DE_RECORRI-

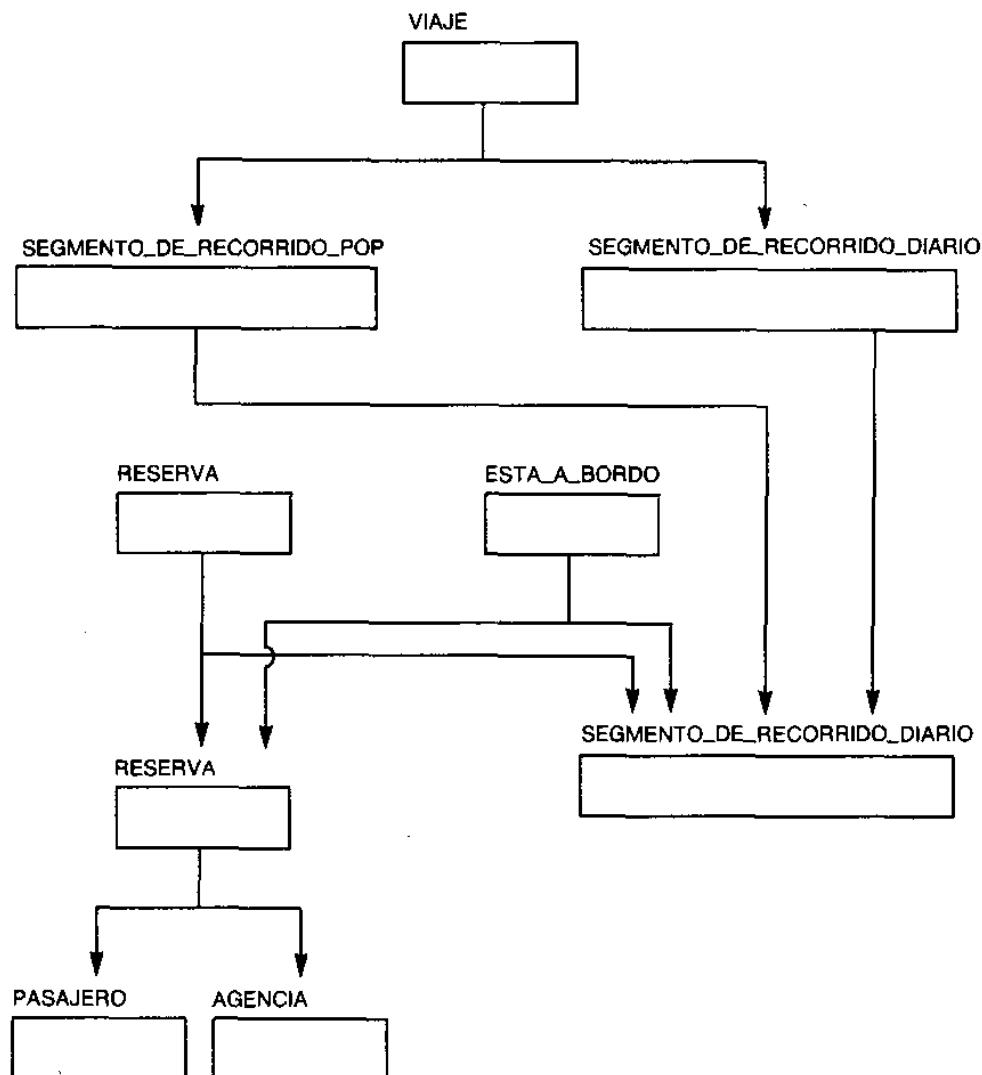


(a) al final del paso 5

Figura 14.13. Resultado intermedio de la correspondencia de la base de datos del caso de estudio con un esquema jerárquico.

DO_DIARIO, que tiene cuatro interrelaciones padre-hijo incidentes. Por la naturaleza de esta aplicación, es perfectamente razonable considerar este tipo de registros copiado, o mejor dicho dividido, en dos tipos de registros denominados SEGMENTO_DE_RECORRIDO_DIARIO_POP y SEGMENTO_DE_RECORRIDO_DIARIO_ORD, representando los dos tipos distintos de segmentos del recorrido diario (cuyas ocurrencias son distintas). Estos se colocan naturalmente debajo de SEG_DE_RECORRIDO_POP y SEG_DE_RECORRIDO_ORD, respectivamente. Se crean tipos de registros vínculos CL₁ y CL₂ (que representan a cliente), debajo de RESERVA y ESTA_A_BORDO, con CLIENTE como tipo de registros padre virtual. Luego se definen dos copias secundarias de SEGMENTO_DE_RECORRIDO_DIARIO debajo de RESERVA y ESTA_A_BORDO, y se definen los tipos de registros SEGMENTO_DE_RECORRIDO_DIARIO_POP y SEGMENTO_DE_RECORRIDO_DIARIO_ORD como sus padres virtuales. El esquema resultante final se muestra en la figura 14.14 y consta de cuatro jerarquías interconectadas por las interrelaciones virtuales padre-hijo.

Respecto a las situaciones especiales que se mencionaron en el apartado 14.2.3, no hay atributos compuestos o polivalentes que sea preciso tratar.



(b) después de hacer a RESERVA y ESTA_A_BORDO tipos de registro raíz

Figura 14.13.Bis Resultado intermedio de la correspondencia de la base de datos del caso de estudio con un esquema jerárquico (*continuación*).

SEGMENTO_DE_RECORRIDO_DIARIO hereda la clave entera, el trío (NUMERO_VIAJE, FECHA, NUMERO_SEGMENTO). Tampoco hay que tratar con ninguna interrelación recursiva. Así, no son necesarias modificaciones posteriores del esquema jerárquico lógico de la figura 14.14.

14.4.2. Correspondencia de operaciones con el modelo jerárquico

Se considera la correspondencia de tres consultas de navegación del caso de estudio (las mismas analizadas en el capítulo 13) con el DML jerárquico corres-

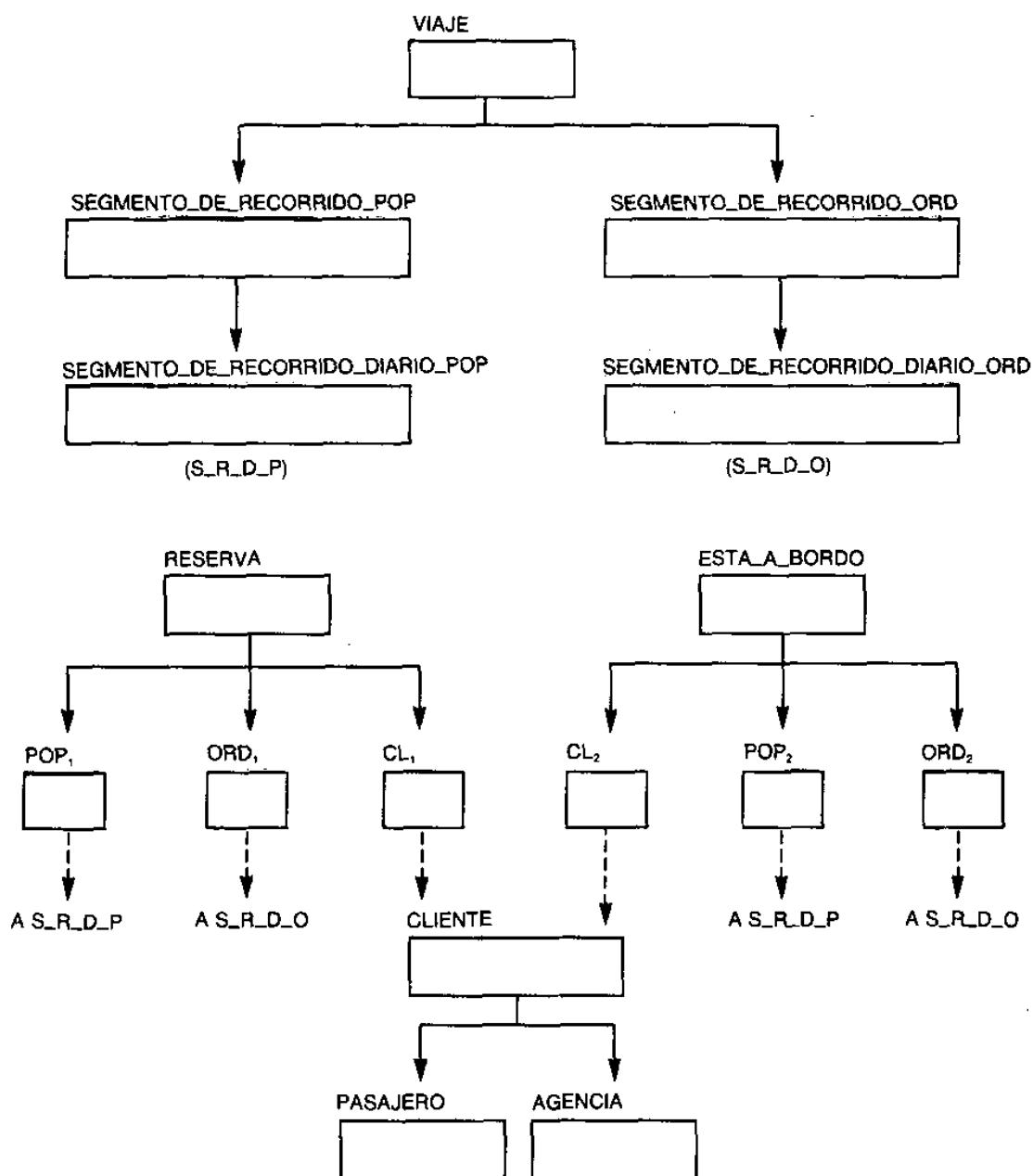
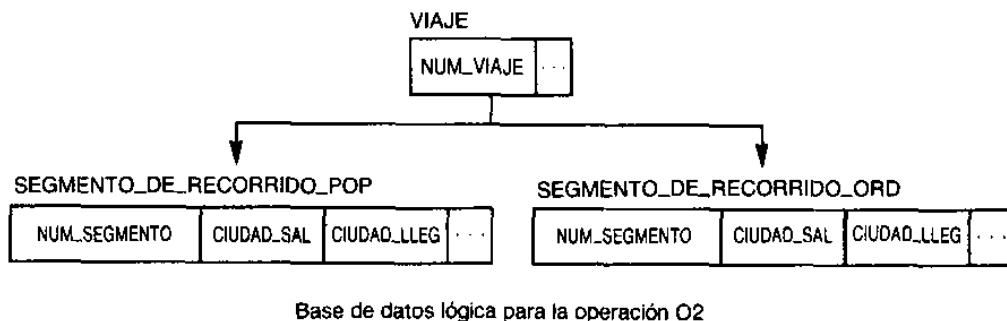


Figura 14.14. Esquema lógico jerárquico del caso de estudio.

pondiente. Para cada consulta es necesario definir una base de datos lógica (usando la terminología IMS) según la cual se ejecute la consulta. La base de datos lógica es una nueva jerarquía definida a partir de los tipos de registros existentes de múltiples jerarquías, como se ilustra en el apartado 14.2.5. Para propósitos de procesamiento, la base de datos lógica es tratada como si fuera una base de datos jerárquica por sí misma.

Para la operación de recuperación (O2), véase la base de datos lógica de la



```

leer ($S,$LL)/* $S es ciudad de salida, $LL es ciudad de llegada */
%GET UNIQUE VIAJE/* acceso al primer registro de viaje en la base de datos */
comenzar
mientras DB_STATUS=0 hacer
    comenzar
        %GET NEXT WITHIN PARENT SEGMENTO_DE_RECORRIDO_POP WHERE
            CIUDAD_SALIDA=$S, CIUDAD_LLEGADA=$LL
        /* acceso al primer segmento de recorrido popular del viaje actual con las ciudades requeridas de salida y llegada, si existe */
        escribir (NUM_VIAJE, NUM_SEGMENTO)
        terminar
    mientras DB_STATUS=0 hacer
        comenzar
            %GET NEXT WITHIN PARENT SEGMENTO_DE_RECORRIDO_ORDINARIO WHERE
                CIUDAD_SALIDA=$S, CIUDAD_LLEGADA=$LL
            acceso al próximo segmento de recorrido ordinario del viaje actual con las ciudades requeridas de salida y llegada, si existe */
            escribir (NUM_VIAJE, NUM_SEGMENTO)
            terminar
        %GET NEXT VIAJE
        /* seleccionar próximo viaje; el ciclo termina cuando no hay más viajes */
        terminar
        /* Obsérvese que un viaje puede contener varios segmentos de recorrido para un par de ciudades dado */
    
```

Figura 14.15. Especificación en DML jerárquico de la operación O2 con la base de datos del caso de estudio.

figura 14.15. Inicialmente, la operación GET UNIQUE se usa para localizar la *primera* ocurrencia de viaje en la base de datos. Las ciudades de salida y llegada introducidas al programa de aplicación se usan como condición en la operación de GET NEXT WITHIN PARENT para encontrar la primera ocurrencia calificada del registro SEGMENTO_DE_RECORRIDO_POP. El GET NEXT WITHIN PARENT se utiliza para buscar a través de todas las ocurrencias de SEGMENTO_DE_RECORRIDO_POP debajo de un viaje dado. Cada SEGMENTO_DE_RECORRIDO_POP que coincide con el par de ciudades dado es presentado con una combinación de NUM_VIAJE y NUM_SEGMENTO. Suponiendo que pueda haber segmentos de recorrido ordinario para el mismo par de ciudades, también se buscan las ocurrencias del registro SEGMENTO_DE_RECORRIDO_ORDINARIO debajo de ese viaje.

Obsérvese que la secuencia SEGMENTO_DE_RECORRIDO_POP seguido por SEGMENTO_DE_RECORRIDO_ORD está gobernada por el ordenamiento de izquierda a derecha de estos segmentos en el esquema de la base de datos. Después que buscar exhaustivamente en un VIAJE, nos trasladamos al próximo VIAJE hasta alcanzar el final de toda la base de datos. Así, en esta consulta se examina la base de datos entera viaje por viaje.

A continuación, en la figura 14.16 se muestra una operación de inserción (O4) para crear un cliente nuevo. Primero se inserta un registro CLIENTE con los atributos comunes de NUM_CLIENTE, NOMBRE y TELEFONO. Dependiendo de que TIPO_CLIENTE sea «PAS» o «AGE», un registro PASAJERO o AGENCIA se creará e insertará debajo de la misma ocurrencia de CLIENTE. Obsérvese que en contraste con el modelo de redes, aquí no hay necesidad de conectar el registro hijo con el padre; el sistema lo inserta automáticamente debajo del padre actual.

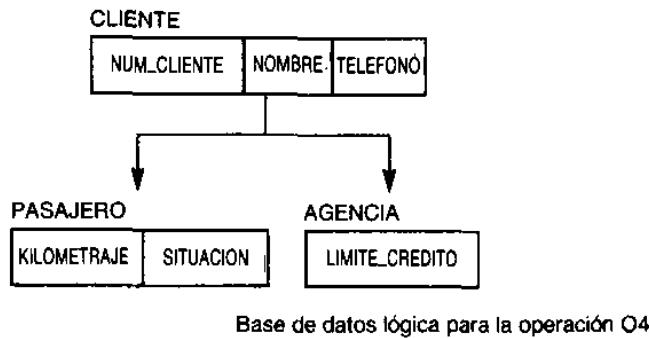
Finalmente, la eliminación de una reserva existente se muestra en la operación O6 con la base de datos lógica de la figura 14.17. Obsérvese que se puede llegar al registro original RESERVA a través de los segmentos populares o a través de los ordinarios; por tanto, se usa dos veces con nombres diferentes, RESERVA_P y RESERVA_O en la base de datos lógica. Se localiza el registro de reserva apropiado con el número de viaje, número de segmento y fecha dados recorriendo hacia abajo la jerarquía. Primero se prueba el brazo izquierdo de la jerarquía. Utilizamos GET HOLD NEXT WITHIN PARENT (conseguir y retener el siguiente dentro del parent) para tener acceso a la reserva de manera que su carácter de actualidad sea mantenido y pueda ser borrada después. Si se encuentra una reserva, se imprime la información pertinente de todos los registros antepasados así como el registro de la reserva en un diario de borrado; luego se borra la reserva con la operación DELETE y se sale.

Si la búsqueda anterior falla, se continúa buscando en el brazo derecho de la jerarquía y se repite el proceso. Si la reserva ya ha sido encontrada, no nos molestamos en examinar los segmentos de recorrido diario ordinario innecesariamente. Es importante observar que los segmentos populares y sus reservas se buscan primero, añadiendo eficiencia al procesamiento.

14.5. Retroingeniería de esquemas jerárquicos a esquemas ER

Como un esquema jerárquico es una forma de esquema de redes, el procedimiento que se ha descrito para la correspondencia de esquemas de redes con esquemas ER es válido aquí. El único preprocesamiento que se necesita es convertir primero el esquema jerárquico dado en un esquema de redes. Esto se hace como sigue:

1. Convertir cada interrelación padre-hijo en un tipo de conjuntos con nombre, como en el modelo de redes.
2. Convertir cada interrelación padre-hijo virtual en un tipo de conjuntos con



```

leer ($CLIENTE.TIPO)
si ($CLIENTE.TIPO)='PAS' entonces
  comenzar
  leer ($CLIENTE.TIPO)
  PASAJERO.KILOMETRAJE:=$KIL
  PASAJERO.SITUACION:=$SIT
  terminar
  si-no
  comenzar
  leer ($NUMC, $NOMBRE, $TEL, $CREDITO)
  AGENCIA.LIMITE_CREDITO:=$CREDITO
  terminar;

CLIENTE.NUM_CLIENTE:=$NUMC
CLIENTE.NOMBRE:=$NOMBRE
CLIENTE.TELEFONO:=$TEL

/* aquí, $NUMC, $NOMBRE, $TEL son valores de entrada válidos para un cliente, $KIL, $ESTADO se aplican a
un pasajero individual, mientras que $CREDITO se aplica a una agencia. $CLIENTE.TIPO= 'PAS' o
$CLIENTE.TIPO='AGE' determina si un cliente es un pasajero o una agencia. Se supone que las variables del
área de trabajo para CLIENTE, PASAJERO y AGENCIA tienen los mismos nombres. */

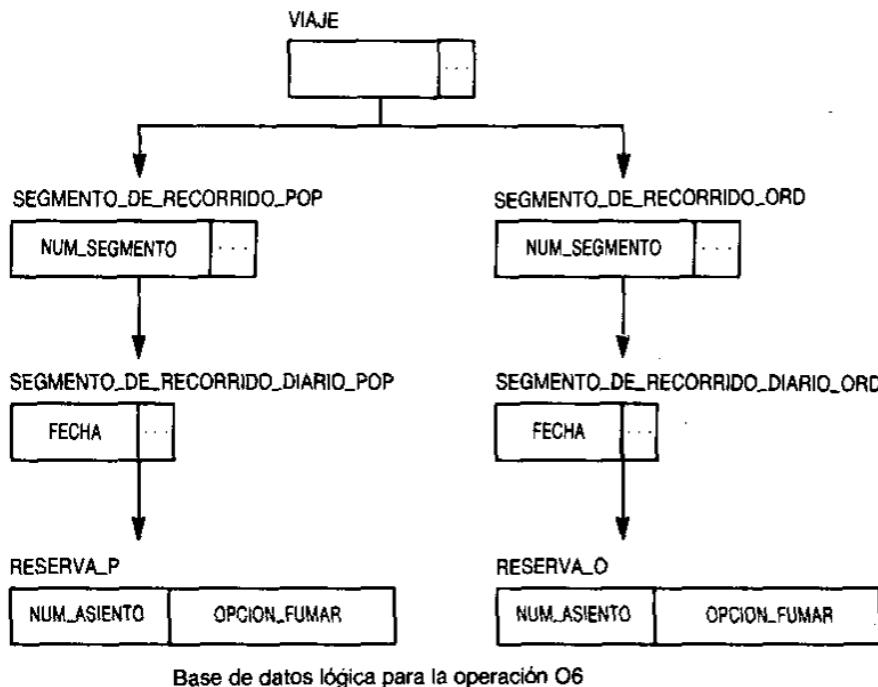
INSERT CLIENTE
si $CLIENTE.TIPO='PAS' entonces
  comenzar
  INSERT PASAJERO
  terminar
  si-no si CLIENTE.TIPO='AGE' entonces
    comenzar
    INSERT AGENCIA
    terminar;
  
```

Figura 14.16. Especificación en DML jerárquico de la operación O4 con la base de datos del caso de estudio.

nombre con el padre virtual como propietario y el hijo virtual como miembro, como en el modelo de redes.

3. Eliminar los registros redundantes, si hay alguno.

Después de ejecutar estos tres pasos, se está en condiciones de aplicar el procedimiento del apartado 13.5 al resultado. Como ejemplo, se ha hecho corres-



```

leer ($V,$S,$F)/* $V es número de viaje, $S es número de segmento, $F es fecha */
BANDERA_SEG_DIARIO_ENCONTRADO:=0/* bandera para indicar que se halló un segmento*/
%GET UNIQUE VIAJE WHERE NUM_VIAJE=$V
/* acceso al registro de viaje con el número de viaje dado */
si DB_STATUS=0 entonces
  comenzar
    %GET UNIQUE SEGMENTO DE RECORRIDO POP WHERE NUM_SEGMENTO=$S
    SEGMENTO_DE_RECORRIDO_DIARIO_POP WHERE FECHA=$F
    mientras DB_STATUS=0 hacer
      BANDERA_SEG_DIARIO_ENCONTRADO:=1 /* segmento encontrado como segmento popular */
    comenzar
      %GET HOLD NEXT WITHIN PARENT RESERVA_P
      escribir DIARIO_BORRAR_RESERVA
      /* diario para auditoría de las reservas borradas */
      %DELETE RESERVA_P
      /* borrar cada reserva encontrada */
    terminar
  terminar
  si BANDERA_SEG_DIARIO_ENCONTRADO=1 entonces salir
  si-no
    comenzar
      %GET UNIQUE SEGMENTO DE RECORRIDO ORD WHERE NUM_SEGMENTO=$S
      SEGMENTO_DE_RECORRIDO_DIARIO_ORD WHERE FECHA=$F
      mientras DB_STATUS=0 hacer
        comenzar
          %GET HOLD NEXT WITHIN PARENT RESERVA_O
          escribir DIARIO_BORRAR_RESERVA
          /* diario para auditoría de las reservas borradas */
          %DELETE RESERVA_O
          /* borrar cada reserva encontrada */
        terminar
      terminar
      si-no
        escribir «no se encontró el viaje»
    salir
  
```

Figura 14.17. Especificación en DML jerárquico de la operación O6 con la base de datos del caso de estudio.

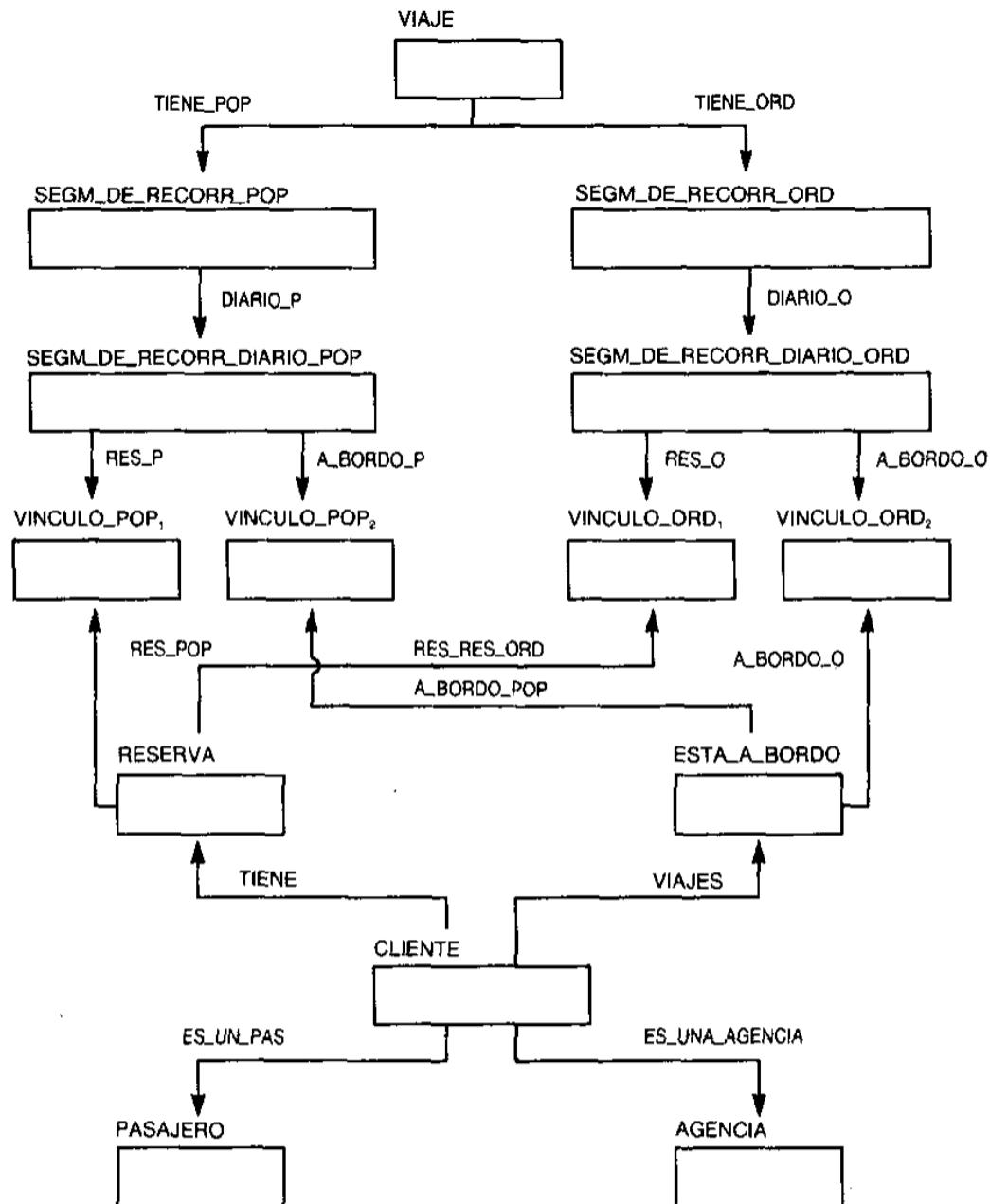


Figura 14.18. Conversión del esquema jerárquico de la figura 14.14 en un esquema de redes para la base de datos del caso de estudio.

ponder el esquema de la figura 14.14 con un esquema de redes, usando los tres pasos anteriores. El resultado se muestra en la figura 14.18. Obsérvese que los registros CL_1 y CL_2 han sido eliminados. Se deja al lector que convierta este esquema en uno ER (véase el ejercicio 14.9).

14.6. Resumen

Este capítulo introduce los conceptos básicos del modelo de datos jerárquico, incluidos los de tipos de registros y tipos de interrelaciones padre-hijo. En concreto, se analizan las propiedades de los esquemas jerárquicos, se muestra cómo una base de datos está formada por árboles de ocurrencia jerárquicos y cómo se almacenan linealmente. La noción de claves no está bien definida en el modelo jerárquico; los registros no tienen necesariamente claves. Se comentan las restricciones de integridad en términos de las propiedades de los esquemas jerárquicos.

No se considera como estándar ninguna versión en particular del modelo jerárquico y del lenguaje de manipulación de datos. Sin embargo, el modelo se conoce mejor en términos del conjunto de funciones comúnmente usadas en el sistema IMS de IBM. Por tanto, se hizo referencia a IMS siempre que fue posible, pero manteniendo el análisis en un nivel general.

Para que el modelo jerárquico gestione interrelaciones de muchos a muchos, se introduce el concepto de tipos de interrelaciones virtuales padre-hijo, los cuales se denominan *interrelaciones lógicas* en IMS. Esta idea permite múltiples jerarquías que puedan estar interrelacionadas, dando lugar a estructuras tipo redes. Una aplicación puede hacer referencia a una *vista*, que es una subjerarquía con un subconjunto de registros de múltiples jerarquías. IMS requiere que cada aplicación procese una vista específica.

A continuación se expuso el diseño lógico de una base de datos jerárquica a partir de un esquema conceptual ER. Esto implica la correspondencia no sólo de los datos sino también de las operaciones y consultas. Se describió una metodología general para hacer corresponder esquemas considerando la correspondencia de entidades con registros y de interrelaciones de uno a uno y de uno a muchos con interrelaciones padre-hijo. Para las interrelaciones de muchos a muchos y n-arias es necesario establecer un tipo de registros aparte y tener interrelaciones padre-hijo apropiadas que apunten *hacia* este registro o *desde* él. Se intenta la correspondencia convirtiendo primero el esquema en una estructura tipo redes, la cual entonces se adapta al modelo jerárquico usando interrelaciones virtuales padre-hijo. Se presenta esto escogiendo padres primarios o creando copias duplicadas de los registros con múltiples interrelaciones incidentes. *Después* del procedimiento de correspondencia general se consideraron situaciones de correspondencia especial que implican atributos compuestos y polivalentes, identificadores externos, e interrelaciones recursivas.

Se mostraron ilustraciones de operaciones de correspondencia de esquemas de navegación ER con el lenguaje de manipulación de datos jerárquico. Esto último se hizo tomando como modelo el lenguaje DL/1 de IMS. Para el caso de estudio, se mostró la correspondencia de los tres esquemas de navegación representativos del capítulo 10, que implican una búsqueda, una inserción y una eliminación, con el lenguaje de manipulación de datos jerárquico.

Finalmente se trató el problema de la retroingeniería desde un esquema je-

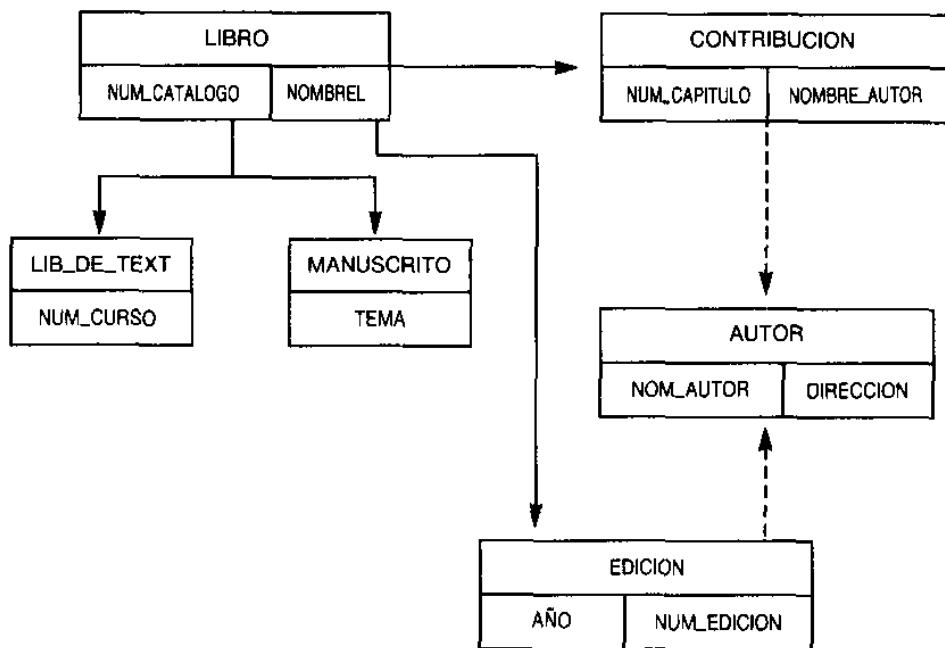


Figura 14.19. Esquema jerárquico de la base de datos de una biblioteca.

rárquico dado a un esquema conceptual abstracto en el modelo ER. Este proceso se realiza fácilmente transformando primero el esquema en otro de redes. Como las bases de datos jerárquicas han estado en uso al menos las dos últimas décadas, una gran cantidad de datos reside en los sistemas jerárquicos sin un diseño adecuado de bases de datos. En la mayoría de las organizaciones, un entendimiento conceptual de estas bases de datos existentes es extremadamente útil antes de la implantación de nuevas aplicaciones o de la conversión de esas bases de datos en DBMS relacionales u orientados a objetos. Los esquemas conceptuales ER pueden servir como interpretación intermedia. Primero se desarrolla la estructura de redes y luego se aplica el procedimiento del apartado 13.5 para hacerla corresponder con un esquema ER. El procedimiento tiene que apoyarse en información adicional respecto a las interrelaciones conjunto-subconjunto, restricciones de cardinalidad, etc., suministrada por el diseñador.

Ejercicios

- 14.1. Considere la base de datos de proyectos de investigación de la figura 2.35. Aplique la metodología descrita en el apartado 14.2 y transforme el esquema lógico ER en un esquema jerárquico.
- 14.2. Considere la base de datos universitaria de la figura 2.33. Sustituya la jerarquía de generalización introduciendo interrelaciones entre las entida-

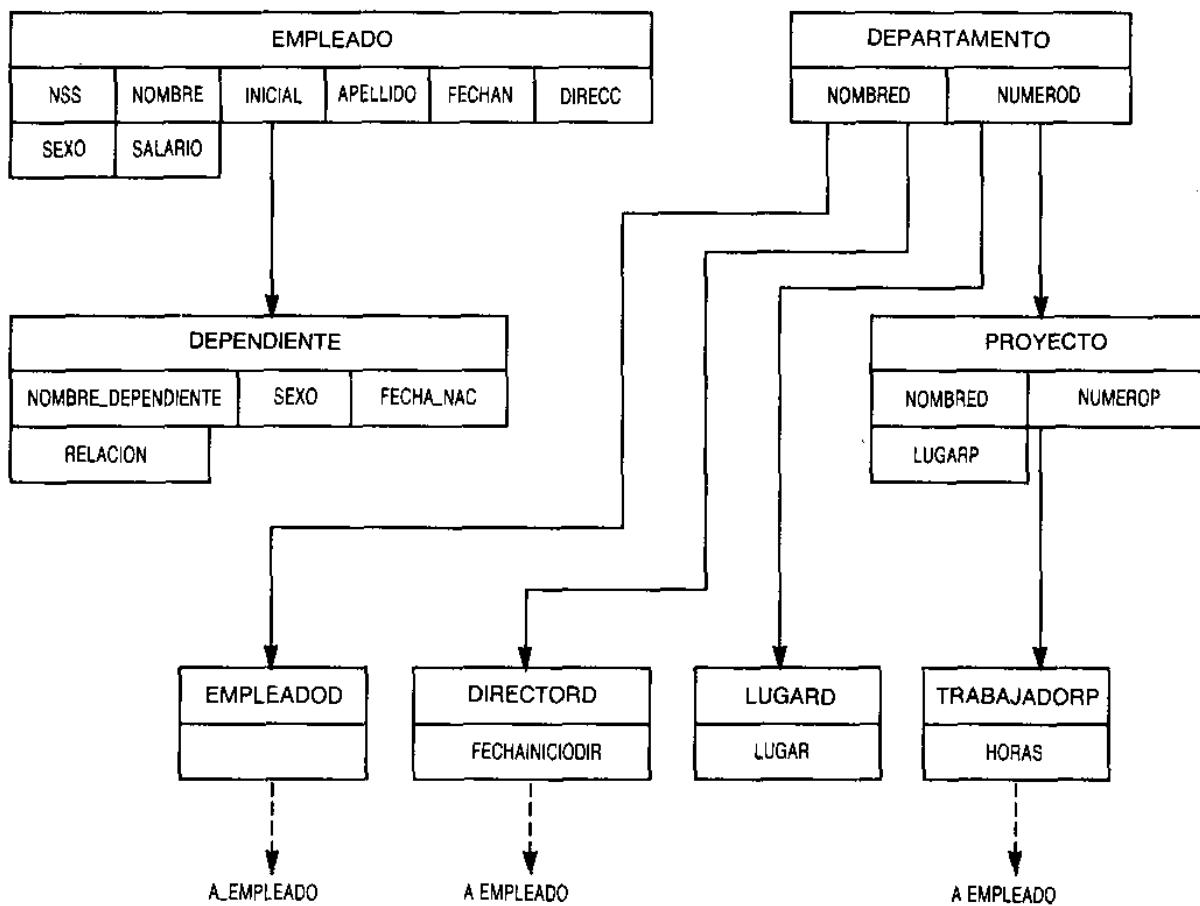


Figura 14.20. Esquema jerárquico de la base de datos de una compañía.

des superconjunto y subconjunto. Aplique la metodología descrita en el apartado 14.2 y transforme el esquema lógico ER en un esquema jerárquico.

- 14.3. Repita el ejercicio 2 con la base de datos de fútbol de la figura 2.34.
- 14.4. Considere el diagrama ER de una base de datos de los departamentos de tráfico y parques de una ciudad (Fig. 12.21).
 - a) Dibuje esquemas de navegación ER para las siguientes consultas (esta parte es la misma que el ejercicio 12.4).
 1. Listar todos los nombres de calles de una ciudad dada.
 2. Listar todas las intersecciones de la calle Main de la ciudad de Gainesville, Georgia.
 3. Listar todos los parques de la ciudad de Gainesville, Florida.
 4. Listar todos los parques localizados de la calle Huron de Ann Arbor, Michigan.
 - b) Convierta el esquema ER de la figura 12.21 en una base de datos jerárquica.

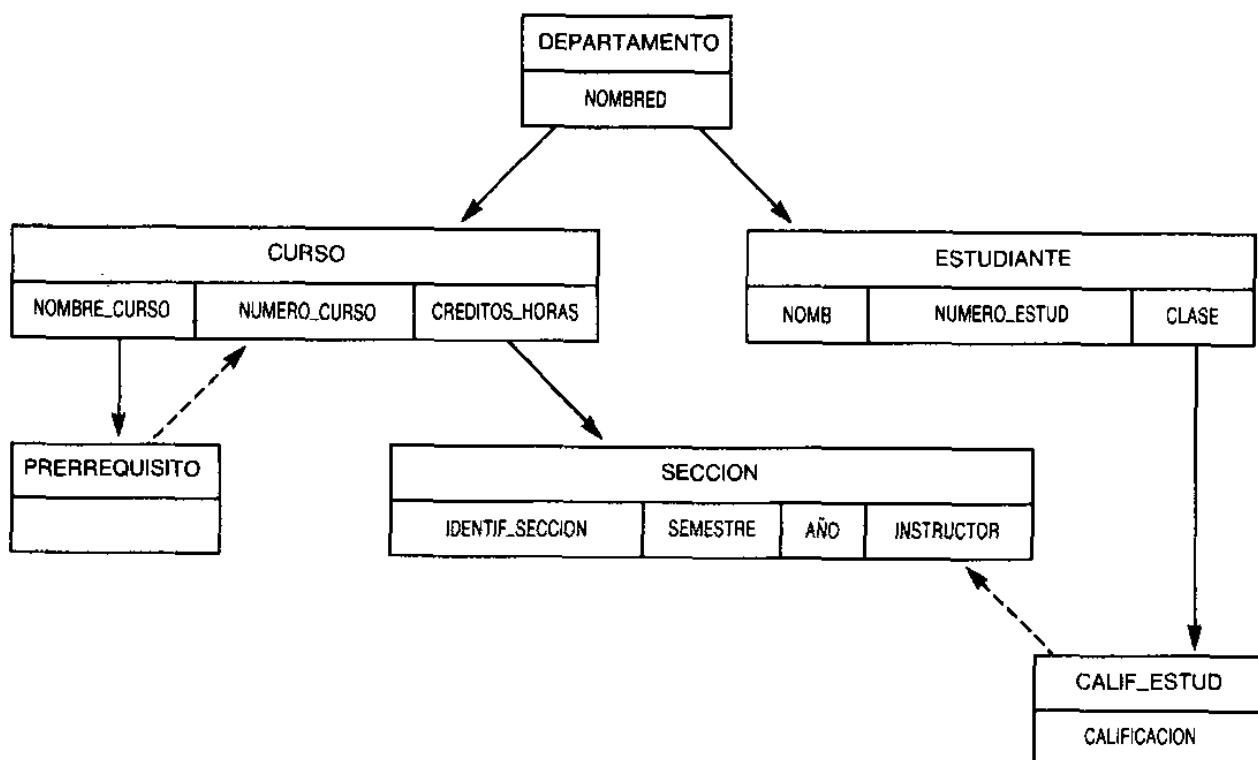


Figura 14.21. Esquema jerárquico de la base de datos de una universidad.

- c) Convierta sus esquemas de navegación a un lenguaje de manipulación de datos jerárquico similar al que se usó en los apartados 14.3 y 14.4.2.
- 14.5. Considere el esquema ER de la base de datos del aeropuerto y vuelos de la figura 12.22. Conviértalo en esquema jerárquico. Haga las suposiciones que necesite durante la correspondencia y establezcalas claramente.
- 14.6. Considere el esquema jerárquico de la base de datos de una biblioteca (Fig. 14.19). La base de datos se explica por sí misma. Aplicando la metodología del apartado 14.5, haga corresponder el esquema con un esquema ER; obsérvese que los libros de texto y los manuscritos son subclases de libros. Verifique que sus resultados correspondan con los de los ejercicios 12.7 y 13.7. Si no es así, determine las razones de cualquier discrepancia.
- 14.7. Considere la base de datos jerárquica mantenida por el departamento de personal de una compañía (Fig. 14.20). El significado de la base de datos es obvio. Suponga que NSS en EMPLEADO, NUMEROD en DEPARTAMENTO y NUMEROP en PROYECTO son claves. Suponga que NOMBRE_DEPENDIENTE es el identificador interno de un dependiente que no es único. Siga la metodología del apartado 14.5 para hacer corresponder el esquema anterior con un diagrama ER. Compare éste con el resultado del ejercicio 12.8. En caso de haber diferencias, vea si puede achacarlas a la interpretación de los esquemas respectivos.

- 14.8. Parta de la base de datos jerárquica de control de pedidos de la figura 14.8 y aplique el procedimiento de retroingeniería para convertirla en un esquema ER. Compare el resultado final con el esquema ER de la base de datos de control de pedidos de la figura 13.7.
- 14.9. Considere la base de datos jerárquica de la figura 14.14 y su correspondencia con un esquema de redes en la figura 14.8. Con su conocimiento del caso de estudio, aplique el procedimiento de retroingeniería para convertirlo en un esquema ER. Compare el resultado final con el esquema ER al que se llegó en la figura 11.15.
- 14.10. Partiendo del esquema jerárquico de una base de datos universitaria de la figura 14.21, aplique el procedimiento de retroingeniería para convertirlo en un esquema ER. Haga las suposiciones que necesite durante la correspondencia y establezcalas claramente.

Bibliografía

W. McGee, «The Information Management System IMS/VS. Part 1: General Structure and Operation», *IBM System Journal*, 16, núm. 2, junio de 1977.

D. Tsichritzis y F. Lochovsky, «Hierarchical Data-Base Management: A Survey», *ACM Computing Surveys*, 8, núm. 1, marzo de 1976, 67-103.

Estas son de las primeras publicaciones sobre el modelo jerárquico. La primera describe el modelo tal como fue implantado en el sistema IMS. La segunda describe el modelo jerárquico de manera más general, pero todavía influido por IMS.

Si se desea estudiar las características básicas del modelo jerárquico y del lenguaje de tratamiento de datos para el modelo jerárquico, véanse los libros de Elmasri y Navathe; Date; Korth y Silberschatz, y Ullman listados en la bibliografía del capítulo 1. Elmasri y Navathe ofrecen un tratamiento general del modelo jerárquico en el capítulo 10 y una descripción específica de IMS en el capítulo 23.

D. Bjoerner y H. Lovengren, «Formalization of Database Systems and a Formal Definition of IMS». En *Proc. International Conference on Very Large Databases*, Ciudad de México, 1982.

IMS/VS General Information Manual, impreso IBM núm. GH20-1260.

IMS/VS Database Administration Guide, impreso IBM núm. SH20-9025.

D. Knapp y J.F. Leben, *IMS Programming Techniques*, 2a. ed., Van Nostrand Reinhold, 1986.

Estas referencias describen principalmente el sistema IMS. El libro de Knapp y Leben describe los conceptos básicos así como detalles de programación en DL/1 y algunos rasgos avanzados de IMS, incluidos las comunicaciones (DC) y Fast Path de IMS. El trabajo de Bjoerner intenta una formalización del modelo de datos IMS. Los dos manuales *IMS/VS* son muestras representativas de manuales de IMS.

System 2000 Language Specification Manual for the DEFINE language, SAS, Inc., núm. 1010.

System 2000 Language Specification Manual for the COBOL Programming Language Extension (PLEX), SAS, Inc., núm. 1020.

Estos textos son representativos de los manuales para otro sistema jerárquico popular, System 2000.

- S.R. Dumpala y S. K. Arora, «Schema Translation Using the Entity-Relationship Approach». En P. Chen, ed., *Entity-Relationship Approach to Information Modeling and Analysis*, North Holland, 1983.
- S.B. Navathe y A. Cheng, «A Methodology for Database Schema Mapping from Extended Entity-Relationship Models into the Hierarchical Model». En P. Chen, ed., *Entity-Relationship Approach to Software Engineering*, Elsevier Science (North-Holland), 1983, 223-48.
- H. Sakai, «A Unified Approach to the logical Design of a Hierarchical Data Model». En P. Chen, ed., *Entity-Relationship Approach to Systems Analysis and Design*, North-Holland, 1980, 61-74.

El trabajo de Dumpala y Arora es la primera referencia que considera la correspondencia de ER a jerárquico, así como la correspondencia inversa de jerárquico a ER. El enfoque está limitado al modelo ER básico y no es completo. El trabajo de Sakai tiene en cuenta la correspondencia «óptima» de ER a jerárquico considerando las transacciones. Navathe y Cheng extienden el trabajo de Sakai para incluir interrelaciones superconjunto-subconjunto y generalizaciones en la entrada y considerar las interrelaciones virtuales padre-hijo en los esquemas objetivo.

- S.B. Navathe y A. M. Awong, «Abstracting Relational and Hierarchical Data with Semantic Data Model», En S. March, ed., *Proc. Sixth International Conference on Entity-Relationship Approach*, Nueva York, North-Holland, 1987.
- J. Winans y K.H. Davis, «Software Reverse Engineering from a Currently Existing IMS Database to an Entity-Relationship Model». En H. Kangassalo, ed., *Proc. Ninth International Conference on Entity-Relationship Approach*, Lausanne, Suiza. North-Holland, 1990.

El primer trabajo trata el problema de la correspondencia inversa de una base de datos relacional o jerárquica a un esquema ER ampliado con categorías para tratar las generalizaciones y las interrelaciones conjunto-subconjunto. Amplía el trabajo de Dumpala y Arora y ofrece un procedimiento general, paso por paso, para la correspondencia inversa. El segundo trabajo se dirige específicamente a la retroingeniería de bases de datos de IMS.

- C. Delobel, «Normalization and Hierarchical Dependencies in the Relational Data Model», *ACM Transactions on Database Systems*, 3, núm. 3, septiembre de 1978, 201-22.
- E. Lien, «Hierarchical Schemata for Relational Databases», *ACM Transactions on Database Systems*, 6, núm. 1, marzo de 1981, 48-69.
- El trabajo de Delobel desarrolla el concepto general de dependencia jerárquica. Los dos trabajos estudian la manera de descubrir estructuras jerárquicas a partir de un conjunto dado de interrelaciones.

Herramientas de diseño de bases de datos

Este capítulo examina el fenómeno relativamente reciente de las herramientas automatizadas y los entornos de apoyo para el diseño de bases de datos. Los tres primeros apartados ofrecen un marco general. El apartado 15.1 ofrece una perspectiva general sobre las herramientas para ingeniería de software asistida por computador (herramientas CASE). El apartado 15.2 destaca las características deseadas en un sistema de diseño de bases de datos. Estas características incluyen: 1) una interfaz «habitável» para el usuario; 2) gran cobertura en las áreas de diagramas, herramientas, interfaces y enfoques de diseño; 3) un conjunto robusto e integrado de herramientas; 4) características para el seguimiento histórico de la metodología y el diseño, incluidos la propagación de cambios y el acceso compartido a los diseños, y 5) una arquitectura que sea extensible en las áreas de representaciones de diseño, herramientas, interfaces externas y metodologías. Al no ofrecer siempre las herramientas existentes un apoyo perfecto a una metodología dada, el apartado 15.3 lista estrategias para superar esta «falta de concordancia» respecto a las representaciones de diseño, los conjuntos de herramientas y el apoyo metodológico.

El apartado 15.4 empieza con una arquitectura de referencia de alto nivel para los sistemas de diseño, y luego introduce al lector dentro de varias herramientas de diseño conceptual. El apartado 15.5 continúa con dos herramientas comunes de diseño lógico. Las descripciones de las herramientas son más bien generales, pero deberán dar al lector una idea clara de qué hacen las herramientas automatizadas típicas. En los apartados 15.6 al 15.8 se describen 12 interesantes sistemas para el diseño de bases de datos, tanto orientados a la investigación como comerciales, y se incluyen varias pantallas que ilustran sus interfaces con el usuario. Las herramientas comerciales se dividen en dos grandes clases: herramientas orientadas al diseño de bases de datos y herramientas CASE más generales, que incluyen apoyo parcial para el diseño de bases de datos.

El apartado 15.9 resume las propiedades clave de una selección más amplia de herramientas disponibles comercialmente, y el apartado 15.10 expone sus limitaciones. Para finalizar, el apartado 15.11 predice futuras tendencias en el área de las herramientas de diseño.

15.1. Ingeniería de software asistida por computador

En los últimos años, ha habido un auge en la disponibilidad y uso de herramientas automatizadas para el desarrollo de todo tipo de software. La primera generación de herramientas de *ingeniería de software asistida por computador* (CASE, *computer-aided software engineering*) se orientaba hacia la captura y exposición de la información de análisis y diseño necesarias para apoyar el tradicional modelo de «cascada» del desarrollo de software: análisis de los requerimientos, diseño, desarrollo del código, prueba y mantenimiento. Estas herramientas ofrecieron una pizarra en blanco, ideal para el diseño de aplicaciones *a partir de cero*.

La segunda generación de herramientas CASE, de reciente disponibilidad, suele estar integrada en bancos de trabajo para una mayor cobertura del ciclo de vida. Son de naturaleza más gráfica. Su enfoque orientado a los datos, aunado a la generación automática de código y esquemas, ayuda al usuario a evitar la introducción de especificaciones en múltiples niveles. Además, los sistemas CASE de segunda generación incorporan recursos de retroingeniería que ayudan a modernizar las bases de datos y aplicaciones *existentes*.

Las herramientas CASE prometen (pero hasta ahora sólo han cumplido parcialmente) un gran número de beneficios importantes para sus usuarios. Estos incluyen un desarrollo controlado por el usuario, aumentos de la productividad total, un entorno de diseño controlado, documentación automatizada y menor mantenimiento. Una razón de la brecha entre las predicciones y la realidad es que muchas organizaciones hacen poco uso de las metodologías *manuales* para el diseño de bases de datos y el desarrollo de aplicaciones, así que el uso de una metodología *automatizada* implica un gran salto. Otra razón es la falta de concordancia entre las metodologías completas y las herramientas limitadas (esto se describe en el apartado 15.3).

A pesar de que las arquitecturas difieren entre los sistemas CASE, en general tales sistemas abarcan lo siguiente:

1. Una *interfaz para usuario* gráfica y textual, que permite al usuario introducir, examinar y modificar los objetos de diseño y recurrir a herramientas y transformaciones.
2. Un diccionario central de datos y procesos (a veces llamado enciclopedia, o sólo diccionario de datos), que controla y rastrea los objetos de diseño.
3. Un conjunto de herramientas de diseño, que abarca ayudas para diagramación, analizadores de diseño y rutinas para generar prototipos y código.
4. Una *biblioteca de diseños*, tanto genéricos como desarrollados previamente, que sirvan como punto de inicio y fuente de inspiración para el desarrollo de nuevos diseños.

Si se desea más información sobre los sistemas CASE, consultese Gane (1990),

Fisher (1988) y McClure (1989) en la bibliografía. (Todas las referencias en este capítulo son para trabajos listados en la bibliografía de este capítulo.)

15.2. Características convenientes en un sistema de diseño de bases de datos

Ahora se enfocan los detalles del apoyo automatizado para el diseño de bases de datos. Los sistemas para diseñar bases de datos operan principalmente sobre esquemas de datos y de funciones. Estos esquemas deben representarse y tratarse dentro del contexto de una metodología coherente para el diseño conceptual, lógico y físico de bases de datos.

Aunque es posible que el entorno automatizado ideal para el diseño de bases de datos no exista, creemos que resulta útil considerar las características deseadas que debe poseer ese sistema. Algunas provienen de principios generales de los sistemas CASE e interfaces con el usuario; otras se relacionan más con la metodología descrita en los capítulos anteriores. Los puntos de la figura 15.1 pueden servir como lista de verificación para ayudar al lector a evaluar el creciente número de herramientas y sistemas para diseño de bases de datos disponibles comercialmente. En los apartados que siguen se explica con detalle cada punto.

15.2.1. Interfaz habitable para el usuario

El usuario interactúa con el sistema en el nivel de la interfaz para el usuario (UI, *user interface*). La interfaz debe, por tanto, ser coherente y cómoda; debe reforzar y clarificar las tareas del usuario; debe ser, en una palabra, *habitável*. La presentación en pantalla de la información debe ser coherente, de modo que información semejante se localice siempre en el mismo lugar o mediante la misma secuencia de mandatos, independiente del contexto. El diseñador debe controlar tanto el *espacio de pantalla* (tamaño de las ventanas, superposición, redimensionamiento, desplazamiento y borrado) como el *espacio de diagrama* (avance por niveles de detalle y movimiento sobre el diagrama mediante el uso de barras de desplazamiento o de ayudas pictóricas para la navegación). La semántica puede expresarse mediante iconos distintivos, por diferencias en los tipos de líneas, diseños y fondos, y por el uso prudente del color. Las técnicas modernas de UI, como son la posibilidad de abrir múltiples ventanas relacionadas entre sí, los acercamientos para ampliar los detalles, la capacidad de enmascarar áreas indeseables de la ventana o del diagrama y de resaltar rutas o subconjuntos de un diagrama, pueden dar vida a la pantalla, permitiendo al usuario seguir una trayectoria dinámica y reforzada visualmente para la creación, exposición, modificación y transformación de los diseños.

1. Interfaz habitable para el usuario

- a) Presentación coherente en pantalla y paradigmas de interacción
- b) Integración de la interfaz para usuario con el proceso de diseño
- c) Posibilidades de personalización

2. Amplia cobertura

- a) Diagramas: apoyo para los esquemas de datos y funcionales
- b) Herramientas: conjunto completo para abarcar todo el proceso de diseño de bases de datos
- c) Interfaces: con herramientas externas y sistemas de bases de datos
- d) Enfoques de diseño: descendentes, ascendentes, centrífugos y mixtos
- e) Metodologías: grados variables de imposición

3. Conjunto de herramientas robusto e integrado

- a) Algoritmos rigurosos y completos
- b) Orientación hacia la semántica del diseño, no sólo gráficos
- c) Capacidad de analizar las implicaciones de los diseños para el rendimiento
- d) Apoyo para la evaluación comparativa de diseños alternativos
- e) Apoyo para esquemas malos y buenos
- f) Capacidad para solucionar la falta de información
- g) Ayuda para recuperarse de resultados erróneos de las herramientas
- h) Herramientas integradas en función de la arquitectura
- i) Herramientas integradas funcionalmente

4. Seguimiento de la metodología y diseño

- a) Definición basada en reglas de las rutas y restricciones metodológicas
- b) Supervisión de la historia de diseño, de los subconjuntos de diseño y de las versiones objeto del diseño
- c) Propagación de cambios
- d) Acceso compartido a los diseños

5. Arquitectura abierta y extensible

- a) Representaciones nuevas
- b) Herramientas nuevas
- c) Interfaces externas nuevas para el intercambio de información con otros sistemas
- d) Metodologías nuevas

Figura 15.1. Características deseables en un sistema automatizado para diseño de bases de datos.

Los mejores sistemas logran integrar sus interfaces para el usuario y el proceso de diseño. Un buen ejemplo es el árbol de diseño del sistema DDEW de la Computer Corporation of America (Reiner et al., 1986), que presenta gráficamente la historia y el origen de un diseño de base de datos y orienta al diseñador mientras examina y recorre los niveles del diseño y sus versiones.

Otro enfoque de UI, muy apropiado para sistemas que aplican una metodología particular de forma estricta, es ofrecer al diseñador una serie de formularios o plantillas tipo «llenar los espacios en blanco». Por ejemplo, Excelerator solicita toda una pantalla llena de información acerca de cada nuevo campo definido: su nombre, nombres alternativos, definición, formatos de entrada y salida, reglas de edición, valor por omisión, mensaje de solicitud, cabecera de columnas, forma abreviada de la cabecera, fuente y demás.

Algunos sistemas de diseño permiten personalizar la UI, con lo que el dise-

ñador puede trabajar con los estilos de diagramas, conjuntos de mandatos, especificaciones de parámetros de mandatos, y editores con que está familiarizado.

15.2.2. Cobertura amplia

Ningún sistema puede satisfacer las necesidades de todos los usuarios —ni debe tratar de hacerlo— pero la flexibilidad y la amplitud de la cobertura son factores importantes para la utilidad del sistema. Debe existir apoyo de diagramación tanto para **esquemas de datos** (entidades, atributos, interrelaciones, subconjuntos, generalizaciones y consultas del esquema) como para **esquemas funcionales** (procesos, flujos de datos, almacenes de datos e interfaces). Asimismo, son útiles los **diagramas de estructura**, que presentan las relaciones jerárquicas entre los módulos del programa y los **diagramas de descomposición**, que ayudan a desglosar e integrar las descripciones de los sistemas. El usuario debe ser capaz de alternar entre las distintas convenciones diagramáticas comunes (por ejemplo, Yourdon, DeMarco, Gane/Sarson, Martin) a fin de personalizar los rectángulos, líneas y puntas de flecha en la pantalla.

Se necesita un conjunto completo de herramientas para abarcar todo el proceso de diseño de una base de datos, desde el análisis de requerimientos y el diseño conceptual hasta el diseño lógico y físico. Asimismo, se puede clasificar las herramientas por la siguiente taxonomía.

1. Los **editores** permiten la introducción, presentación y modificación de la información en listas, formularios y diagramas. Son especialmente importantes los editores de diagramas; permiten al diseñador añadir objetos y conexiones entre los objetos, redirigir los arcos de conexión, reubicar los objetos y borrarlos.
2. Los **analizadores** realizan análisis, con frecuencia presentados como informes estándar, para que el diseñador actúe según su criterio. Los analizadores pueden verificar la validez de esquemas individuales, comprobar la coherencia en el uso y las definiciones entre los esquemas de datos y los funcionales, detectar violaciones de la normalización, evaluar el rendimiento de un diseño de base de datos bajo una carga específica de transacciones o sugerir índices que se necesitan para un mejor rendimiento de la base de datos.
3. Los **transformadores** producen esquemas nuevos o modificados a partir de los antiguos. Los transformadores pueden dividirse en dos subclases: los que tratan de mantener (y, posiblemente, aumentar) la información presente en un principio y los que toman decisiones y alternativas «inteligentes» que alteran el contenido de información del esquema.
 - a) Los **transformadores de mantenimiento del contenido** comúnmente realizan transformaciones entre modelos de datos, como las corresponden-

cias ER a relacional, ER a jerárquico y ER a redes descritas en los capítulos 12 al 14. Son menos frecuentes los que trabajan en el **nivel de microtransformaciones**, ayudando a lograr minimalidad, legibilidad y normalización, como se describió en el capítulo 6. En algunos sistemas, los transformadores pueden iniciarse en un **modo analítico**, para que el diseñador entienda el impacto de una transformación antes de realizarla. En otros, como el prototipo Physical Designer desarrollado para el DBMS Model 204 en la Computer Corporation of America, el diseñador puede solicitar que el transformador haga concesiones globales (respecto al rendimiento, uso del almacenamiento, complejidad y extensión) en los diseños creados.

- b) Los **transformadores «inteligentes»** generalmente se utilizan en una fase temprana del diseño y toman decisiones tipo mejor conjetura para, rápidamente, crear un resultado editable y utilizable. Se guían, en general, por reglas, y se llaman a veces herramientas heurísticas. Ejemplos serían una herramienta para la integración de vistas, que hace suposiciones razonables en lugar de pedir al usuario que tome un número enorme de decisiones menores sobre un esquema, y una herramienta que sintetiza un esquema ER a partir de dependencias funcionales y definiciones de transacciones simples. Esas herramientas se exponen con más detalle en Rosenthal y Reiner (1989).

No es sorprendente que los conjuntos de herramientas de diseño tengan un valor limitado si no cuentan con interfaces para herramientas externas y sistemas de bases de datos. Los **generadores y cargadores de esquemas** son transformadores que ayudan a exportar e importar a partir de un entorno de sistema externo, usualmente el compilador de DDL o el diccionario de datos de un DDMS. Las transacciones que se definen en un sistema de diseño se pueden traducir al **lenguaje de cuarta generación (4GL)** del sistema de base de datos objetivo (o a cualquier otro lenguaje para desarrollar aplicaciones).

Cobertura amplia significa, asimismo, apoyar una variedad de enfoques de diseño similares a las estrategias descendente, ascendente, centrífuga y mixta descritas en el capítulo 3. En otras palabras, el sistema no debe imponer restricciones rígidas respecto a la parte de un diseño en la que se debe trabajar en un momento determinado, y debe ser tolerante respecto a las distintas cantidades de detalle en los componentes de un diseño en evolución.

En el más alto nivel, un sistema de diseño ideal debe permitir al administrador de bases de datos o al diseñador determinar hasta dónde se debe imponer un control metodológico estricto. Por supuesto, esto no quiere decir que se deba permitir a un usuario construir diagramas malos o absurdos. Más bien, debe haber una gama de opciones disponibles, desde un enfoque estrictamente secuencial de uso de las herramientas, en el que no sea posible elegir las herramientas ni el orden en que se aplican, hasta un enfoque menos estricto, donde las herramientas formen un **banco de trabajo de metodología neutral** que permita múltiples puntos de entrada y secuencias de herramientas.

15.2.3. Conjunto de herramientas robustas e integradas

A pesar de que en las publicaciones teóricas sobre diseño de bases de datos a menudo se hacen suposiciones simplificadas sobre los datos de entrada a los algoritmos, los problemas del diseño práctico no se pueden abordar usando esos supuestos. Se desea, por tanto, un sistema que disponga de herramientas robustas. Por ejemplo, las herramientas no deben exigir que todas las entidades posean llaves de un solo atributo o que todos los nombres de atributos (entre las diferentes entidades) se elijan de manera coherente¹. Las herramientas deben orientarse también hacia la semántica subyacente de los diseños en desarrollo, y no a la capa más superficial de la presentación gráfica. Deben ayudar a evaluar y mejorar la coherencia y corrección de los diseños, además de proporcionar apoyo para la representación esquemática.

Desafortunadamente, un diseño coherente y correcto puede también ser ineficiente e imperfecto. Por ello, es necesario que los conjuntos de herramientas puedan analizar las implicaciones de rendimiento de los diseños que ayudan a crear. Las herramientas deben apoyar la evaluación de ventajas y desventajas entre diseños alternativos, con base en el rendimiento anticipado y otros factores (como son la sencillez o el grado de divergencia respecto a un diseño base). Las herramientas también deben ser tolerantes y permitir esquemas tanto malos como buenos. A veces, un sistema de diseño puede necesitar manipular esquemas que violan la normalización, la no redundancia, la uniformidad de las convenciones de nombres y otras pautas. Tales esquemas pueden ser el resultado de trabajos anteriores o pueden implicar violaciones relacionadas con áreas en las que el rendimiento es vital. Como se expuso en capítulos anteriores, el diseño de bases de datos es un proceso iterativo en el que el diseñador se acerca gradualmente a una representación correcta y completa del diseño en un determinado nivel; por ello, las herramientas también deben ser capaces de hacer frente a la falta de información.

Entre las suposiciones poco realistas que a veces hacen las herramientas, y las entradas incompletas, poco fiables y a veces incorrectas de los usuarios, es inevitable que algunos resultados de las herramientas sean incorrectos. Aunque muchos usuarios tienen una fe implícita en las salidas de las herramientas automatizadas, un buen sistema tiene que facilitar de alguna forma que el usuario detecte y corrija los errores. Por ejemplo, algunos resultados dudosos y conjeturas del sistema se pueden presentar en rojo, para atraer la atención del usuario hacia ellos. Otros problemas se pueden detectar, a veces, cuando el diseñador manipula un objeto erróneo. Los nombres generados por el sistema, que casi nunca serán tan significativos como los proporcionados por el usuario, se pueden marcar con claridad para que el usuario pueda mejorarllos.

El usuario debe sentir que el conjunto de herramientas está integrado. Las

¹ Este último supuesto se conoce como *sujeto de la relación universal* (Maier, Ullman y Vardi, 1984). Dice que los atributos con un significado igual en la realidad (por ejemplo, NSS o FECHA_DE_POLIZA) deben tener nombres idénticos donde quiera que aparezcan en la base de datos.

herramientas integradas respecto a la arquitectura poseen una interfaz común para el usuario y emplean un enfoque uniforme para la manipulación del diagrama, la presentación de la información, el tratamiento de errores y la ayuda. Las **herramientas integradas funcionalmente** sólo requieren que la información sea introducida una vez; no la pierden cuando se cambia de fase o de herramienta de diseño.

15.2.4. Seguimiento de la metodología y el diseño

Conforme el diseñador sigue una metodología, utilizando herramientas y actuando de acuerdo con los resultados, el sistema debe ayudar a rastrear y controlar la secuencia y las interrelaciones de los diseños. Por ejemplo, un sistema podría ayudar al usuario a seguir nuestra metodología de análisis conjunto de datos y funcional (Cap. 9) sugiriendo (o forzando) una alternancia entre los dos tipos de análisis. Durante la integración de los esquemas, podría sugerir un orden para la fusión de los esquemas restantes (pero permitir al usuario ignorarlo).

Asimismo, debe ser fácil para el usuario consultar la historia de diseño de un diagrama o especificación, crear y guardar las versiones de los diseños, y concentrarse en subconjuntos del diseño total. Si se hacen cambios a una representación de diseño, deben propagarse hacia otras de forma controlada (casi siempre utilizando un depósito o diccionario central). Las grandes tareas de diseño que implican a varios diseñadores requieren un *acceso compartido* a los diseños. Un control de **comprobación de entrada/salida** para los diseños puede impedir que los cambios realizados por un usuario sustituyan a los cambios hechos por otro.

Finalmente, las rutas metodológicas permitidas y las restricciones pueden estar definidas por un conjunto de reglas. Esta orientación de sistema experto crea una base reglamentaria que representa las reglas y directrices inherentes de la metodología. Tiene la ventaja de hacer a las metodologías explícitas y fácilmente modificables.

15.2.5. Arquitectura abierta y extensible

Un sistema de diseño de base de datos bien construido debe ser extensible en múltiples direcciones. A menudo se necesitará añadir nuevas representaciones y herramientas para apoyar nuevas metodologías u ofrecer mayor cobertura. Un **sistema abierto** expone suficientes detalles de su funcionamiento para hacer posibles esas adiciones. Esto puede hacerse en un nivel bajo, exponiendo las representaciones internas y las funciones que las manipulan, o a un nivel más alto, construyendo el sistema a partir de objetos y métodos asociados. En el último caso, los objetos y sus métodos pueden hacerse públicos, y un diseñador con experiencia puede añadir también objetos y métodos nuevos.

Como se dijo en el apartado anterior, si las rutas metodológicas se guían por

reglas, el diseñador puede ampliarlas y modificarlas cambiando o aumentando las reglas. Otra ampliación común de un sistema de diseño es la adición de nuevas interfaces externas para intercambio de información con sistemas como DB2, dBase IV, ORACLE y Lotus 1-2-3, o con otros diccionarios y depósitos como el AD/Cycle Repository de IBM y CDD/Plus de DEC. Estas interfaces se llaman con frecuencia **recursos de importación y exportación**.

15.3. Falta de concordancia entre metodologías y herramientas

La llamada **falta de concordancia** entre una metodología y un grupo de herramientas es la correspondencia inexacta entre sus conceptos, enfoques y alcances. En términos más específicos, pueden existir diferencias entre los pasos del diseño, la secuencia de las decisiones, las entradas requeridas, los modelos de datos, las salidas esperadas, los niveles de abstracción y la amplitud de la cobertura del proceso de diseño.

15.3.1. Razones para la falta de concordancia

Sea cual sea la metodología de diseño de bases de datos que se elija, es cosa segura que no la podrá apoyar, de forma exacta y exhaustiva, ningún sistema de diseño. En particular, ningún sistema disponible en la actualidad apoya realmente la metodología descrita en este libro (aunque algunos de los sistemas descritos en los apartados 15.7 y 15.8 se acercan más que otros). Existe una serie de razones para la falta de concordancia entre metodologías y herramientas. Una es que, antes de poder automatizar una metodología, ésta debe estar bien especificada y haber sido comprobada extensamente. Así, las metodologías son, en sí, más ricas que las herramientas automatizadas que van detrás de ellas. Otra es que, para reducir la brecha de aprendizaje y fomentar el compromiso corporativo, las organizaciones prefieren tener en funcionamiento una metodología de diseño manual antes de automatizarla. El resultado es que la metodología se habrá refinado para las necesidades de la organización mucho antes de que llegue una herramienta general de diseño capaz de automatizarla, contribuyendo a la falta de concordancia. Finalmente, los sistemas de diseño automatizados están en una etapa temprana de su desarrollo, como la mayoría de las herramientas CASE. En consecuencia, los gráficos extensos, la flexibilidad metodológica y las capacidades de personalización que permitirían a las herramientas apoyar realmente una metodología específica son técnicamente imposibles todavía.

El resultado final es que probablemente sea necesario usar herramientas automatizadas poco acopladas a la metodología preferida; asimismo, podría faltar apoyo directo para ciertas representaciones, procesos y algoritmos, decisiones y objetivos del diseño.

15.3.2. Cómo resolver la falta de concordancia

A continuación ofrecemos algunas estrategias generales para resolver la falta de concordancia en las áreas de representaciones de diseño, conjuntos de herramientas y enfoque metodológico.

Falta de concordancia 1. Las representaciones pueden utilizar diferente notación y no apoyar diversas convenciones gráficas y detalles de diseño.

Estrategias. La mayoría de las variantes gráficas del modelo ER muestran en esencia la misma información básica (entidades, atributos, interrelaciones, restricciones de cardinalidad, identificación, etc.). Una vez que se entiende una variante y sus convenciones gráficas, las demás no son difíciles de entender. Si faltan ciertas convenciones gráficas o detalles, se pueden compensar con comentarios y anotaciones sobre los objetos o usando convenciones de nombres propias. Por ejemplo, si un sistema no posee una representación gráfica especial para los subconjuntos, se puede añadir el sufijo «sub» a los nombres de las interrelaciones subconjunto, o usar los nombres «subconjunto_1», «subconjunto_2», y así sucesivamente. Si las cardinalidades de las interrelaciones no se rastrean o no se muestran en pantalla, pero se permiten las anotaciones en lenguaje ordinario sobre las interrelaciones, se puede especificar las cardinalidades en una anotación.

Falta de concordancia 2. El conjunto de herramientas puede tener vacíos funcionales y las herramientas pueden hacer más o menos de lo que se espera en un área determinada de diseño.

Estrategias. Aunque un determinado conjunto de herramientas no esté completo, es posible que se pueda enlazar con otros sistemas CASE o con otras herramientas (no integradas) que llenen los vacíos. Esto se puede lograr con flexibilidad en sistemas con buenos recursos de importación/exportación o con una arquitectura abierta. Si las herramientas apoyan un formato intermedio común, puede evitarse una segunda captura de la información del esquema. Las alternativas son hacer ciertos análisis y transformaciones totalmente a mano o intervenir para completar y ampliar las acciones de las herramientas.

Es importante reconocer que ciertas partes de una metodología no pueden automatizarse fácilmente. Se debe tener presentes las reglas, técnicas y pautas de la metodología, aun cuando al parecer las herramientas no las empleen o apoyen específicamente. Por ejemplo, las características de *expresividad, simplicidad y compleción* de los esquemas conceptuales (descritas en el apartado 6.1) son criterios de evaluación subjetivos que una herramienta no puede medir significativamente.

Falta de concordancia 3. El enfoque metodológico encarnado en el sistema

puede ser genérico y no adaptado a las rutas o secuencias iterativas que se deseé seguir.

Estrategias. Realizar un seguimiento del diseño propio, aunque la UI del sistema no esté integrada ni ayude. Específicamente, se puede vigilar el avance propio en el desarrollo conjunto de esquemas de datos y funcionales, y alternar entre el refinamiento de cada uno de ellos. Para la integración de vistas, se puede dar a cada vista de usuario un nombre que sugiera su origen y plantear sobre papel el orden en que se integrarán y los nombres que se darán a los esquemas intermedios resultantes de este proceso. El diseñador mismo debe forzar el cumplimiento de las convenciones de nombres específicas para la metodología o la instalación.

Muchas herramientas no ofrecen apoyo directo para los enfoques de diseño ascendentes, descendentes, centrífugos o mixtos, pero permiten seguir cualquiera de ellos por defecto. Sólo se necesita elegir el punto inicial y tener presente la dirección en que se avanzará. Si las herramientas permiten el uso de diferentes tipos de letra, colores o caracteres especiales, pueden adaptarse para que ayuden en el seguimiento de las acciones metodológicas y del grado de avance. Bajo las estrategias centrífugas y mixtas es casi obligatorio mantener varios esquemas en evolución concurrente.

Incluso armados con un simple editor de diagramas, podemos aplicar las transformaciones descritas en el capítulo 6 para refinar un esquema. Si no hay una biblioteca de esquemas en línea, se puede consultar este libro y otros para ver ejemplos, o estudiar informes sobre bases de datos diseñadas con anterioridad en la organización.

15.4. Herramientas básicas para el diseño conceptual

Este apartado y el siguiente presentan descripciones breves sobre cómo operan las herramientas automatizadas comunes. El objetivo es obtener un conocimiento razonable de qué hacen las herramientas. Aunque están basadas en herramientas automatizadas que hemos desarrollado, usado o estudiado, la mayoría de estas descripciones no están sujetas a sistemas particulares (véase en los apartados 15.6 a 15.9 la exposición de los sistemas de diseño actuales). A fin de preparar el terreno para un análisis de herramientas específicas, empezaremos con una arquitectura de referencia de alto nivel para las herramientas.

15.4.1. Arquitectura de referencia

Capítulos anteriores han ilustrado la secuencia de pasos, métodos, procesos, algoritmos y reglas heurísticas de diseño que se debe usar al diseñar bases de datos. La figura 15.2 es diferente; representa una *visión arquitectónica*, más que

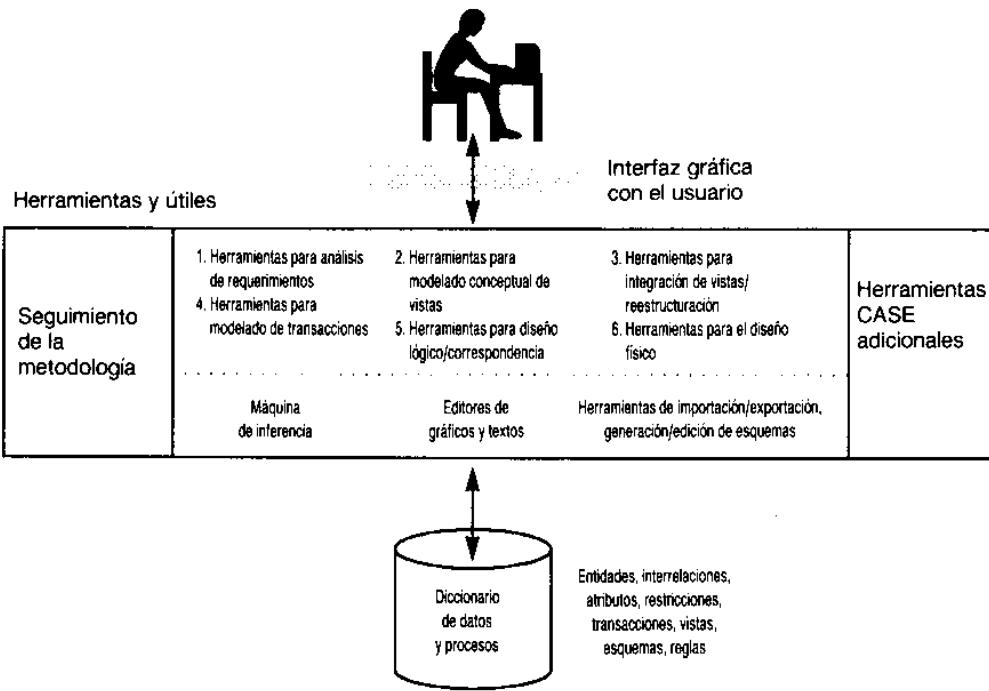


Figura 15.2. Arquitectura de referencia para sistemas automatizados de diseño de bases de datos.

metodológica, de un sistema de diseño. La descomposición en componentes y herramientas individuales corresponde a la estructura de módulos de software que tienen muchos sistemas para automatizar el diseño de bases de datos. Los componentes debajo de la línea punteada son útiles, a menudo compartidas por varias herramientas de más alto nivel. Las herramientas de diseño se aplican por lo regular en el orden 1 a 6; sin embargo, algunas pueden omitirse. Se sugiere tener presente esta **arquitectura de referencia** a medida que se lea en este capítulo acerca de sistemas y herramientas reales.

15.4.2. Editor de diagramas ER

Casi todo sistema automatizado que incorpora el modelo ER posee un editor de diagramas ER. Esta herramienta puede crear un diagrama *a partir de cero*; es posible que su planificador interno de disposición intente minimizar las longitudes de los conectores y los pasos y todavía dejar algún espacio libre para asegurar la legibilidad. El editor trata también de apoyar la *modificación gradual* de los diagramas ER; las entidades e interrelaciones se pueden añadir, borrar, cambiar de nombre y de posición, y se puede reorientar, seguir y subrayar los arcos de conexión. A petición, la herramienta oculta ciertos detalles, como los atributos, y los hace disponibles mediante un acercamiento a una entidad o in-

terrelación. Otros detalles, como las cardinalidades de las interrelaciones, se muestran y editan gráficamente.

Un requisito mínimo de un editor de diagramas ER es que finalmente debe producir diagramas sintácticamente válidos (aunque los pasos intermedios pueden tener, a veces, interrelaciones desconectadas en la pantalla). Un editor «inteligente», orientado hacia la metodología de este libro, permitirá al diseñador solicitar cualquiera de las transformaciones descendentes de los esquemas ilustradas en la figura 3.2, como la expansión de una interrelación para dar una entidad con dos interrelaciones. Asimismo, permitiría reducir temporalmente cualquier subconjunto de un diagrama ER a una caja sencilla, abstracta, para que así el diseñador pueda concentrarse en los detalles del resto del diagrama.

15.4.3. Editor de diagramas de flujo de datos (DFD)

Un editor de DFD apoya la creación y manipulación gráficas de los diagramas de flujo de datos (equivalentes a esquemas funcionales) para ayudar en el análisis funcional. Permite al diseñador añadir, borrar, y cambiar de nombre y de lugar los procesos, flujos de datos, almacenes de datos e interfaces externas. Un sencillo rectángulo de un proceso en un nivel se puede descomponer en un DFD más detallado en el siguiente nivel de detalle y el usuario de la herramienta puede bajar y subir por la jerarquía de niveles detallada de los DFD.

Con frecuencia, los editores de DFD apoyan una operación de transformación conocida como *equilibrio del DFD*, lo que impone la noción de *equivalencia en las fronteras* expuesta en el capítulo 8 (por ejemplo, véase la figura 8.11). El equilibrio del DFD implica comprobar que las entradas y salidas en las fronteras del proceso sean coherentes entre los diferentes niveles. Por ejemplo, si NUM_CLIENTE es una entrada al proceso INTRODUCIR_PEDIDO_CLIENTE en el nivel más alto, NUM_CLIENTE debe, asimismo, aparecer como entrada a un subproceso al descomponerse INTRODUCIR_PEDIDO_CLIENTE en un nivel inferior del DFD. Además, un editor de DFD inteligente ayudaría al usuario a realizar las transformaciones de los esquemas de funciones mostradas en la figura 8.3.

15.4.4. Analizador/integrador de vistas

En el capítulo 5 se ofrece un panorama general de los enfoques de integración de vistas. Esta descripción de herramienta se basa en el sistema DDEW (Reiner et al., 1986), que se describe más adelante en el apartado 15.6.1. La herramienta ayuda a fusionar dos vistas de usuario diferentes en un solo esquema ER. Cuando se aplica de forma recursiva, el proceso resulta en un solo esquema global. En primer lugar, la herramienta detecta conflictos entre las dos vistas e informa de ellos al usuario, como se ilustra en la figura 15.3. En segundo lugar, el usuario resuelve manualmente algunos de los conflictos de los esquemas originales, o todos. En tercer lugar, la herramienta fusiona los esquemas fuente en uno y resuelve cualquier conflicto que haya quedado. Se dan más detalles sobre estos

[Mismos nombres, objetos incompatibles -- necesidad de arreglo]
 Nombre: CARGO. 1.^{er} diagrama: Ent. 2.^o diagrama: Interrel

[Mismos nombres, objetos incompatibles -- necesidad de arreglo]
 Nombre: HABITACION-PACIENTE. 1.^{er} diagrama: Interrel. 2.^o diagrama: Ent.

[Mismos nombres -- los objetos serán fusionados]
 Nombre: OPERACION de tipo: Ent.
 hay conflictos:: diferentes listas de atributos

[Mismos nombres -- los objetos serán fusionados]
 Nombre: SUPERVISOR de tipo: Interrel.
 hay conflictos:: conflictos de cardinalidad

[Mismos nombres -- se sugiere el arreglo]
 Nombre: ESPECIALIDAD. 1.^{er} diagrama: Atr. 2.^o diagrama: Interrel.

[Misma llave -- ¿misma entidad?]
 1.^{er} diagrama: CARGO. 2.^o diagrama: COSTO

[Mismos puntos finales -- ¿misma interrelación?]
 1.^{er} diagrama: ENLACE-PACIENTE. 2.^o diagrama: PERSONAL-MEDICO

Figura 15.3. Parte del informe de conflictos del integrador de vistas DDEW después de comparar dos versiones de un esquema de hospital.

tres pasos más adelante. Todo el proceso de integración de vistas se sigue gráficamente a nivel de esquemas en el árbol de diseño de DDEW.

Primer paso: Detección automática de conflictos. Técnicamente, la herramienta busca **homónimos** (véase [a] y [b] más adelante) y **sinónimos** (véase [c]) en los dos esquemas. En términos prácticos, la herramienta detecta los siguientes conflictos y ambigüedades en potencia:

- Los objetos (entidades, interrelaciones o atributos) podrían tener *tipos diferentes* pero los *mismos nombres* en ambos esquemas (por ejemplo, CARGO y HABITACION-PACIENTE en la figura 15.3). Los conflictos entidad-interrelación se marcan como los más graves; los conflictos que implican atributos no son tan problemáticos (por ejemplo, ESPECIALIDAD).
- Los objetos del *mismo tipo* y con el *mismo nombre* son candidatos para la fusión, pero puede haber *conflictos* entre sus detalles (por ejemplo, OPERACION y SUPERVISOR).
- Los objetos del *mismo tipo* y de *estructura similar* que poseen *nombres diferentes* son candidatos para la fusión (por ejemplo, CARGO y COSTO, ENLACE-PACIENTE y PERSONAL-MEDICO). Por supuesto, muchos casos no pueden detectarse sólo por la semejanza estructural; el diseñador debe in-

formar a la herramienta de forma explícita (mediante el diccionario de datos), de todos los sinónimos conocidos (también llamados **alias**).

Segundo paso: Resolución manual de conflictos. Al examinar el informe de la herramienta después del paso de detección de conflictos, el diseñador cambia nombres y modifica o borra entidades, interrelaciones y atributos según sea necesario para resolver las ambigüedades y los conflictos en los dos esquemas. Los objetos que realmente no sean iguales deberán recibir nombres diferentes. Los conflictos a nivel de atributo no son tan serios, pero las diferencias en la cardinalidad de las interrelaciones pueden indicar formas totalmente distintas de visualizar la base de datos. Por ejemplo, en un esquema los empleados *deben* pertenecer a un departamento; en otro, las entidades EMP no necesitan enlazarse con las DEPTT. El diseñador debe decidir qué semántica es la correcta y ajustar los esquemas en conformidad. Otra estrategia es reconocer cuándo los objetos forman parte de una jerarquía de generalización. El capítulo 5 ofrece detalles amplios sobre la resolución de los conflictos, así que no se volverán a mencionar aquí.

Tercer paso: Fusión automática de los esquemas. La herramienta fusiona objetos que tienen los mismos nombres (o que han sido declarados como sinónimos), resolviendo los conflictos restantes (como las diferencias entre los atributos descriptivos de dos entidades que se fusionan). Luego, añade cualesquier objetos adicionales que haya en los dos esquemas.

15.4.5. Analizador de esquemas ER

Un analizador de esquemas ER busca problemas sintácticos y estructurales en el esquema, como esquemas mal formados o incongruentes. Si bien algunos problemas de nombres los puede detectar de forma dinámica un diccionario de datos y procesos, el analizador de esquemas puede hacer más, dirigiendo la atención del diseñador a esos problemas y a problemas potenciales como éstos:

- Nombres ausentes de entidades, interrelaciones o atributos.
- Nombres que violan las convenciones definidas por el usuario.
- Nombres usados en conflicto con nombres idénticos o semejantes registrados en el diccionario.
- Cardinalidades ausentes en las interrelaciones.
- Entidades sin atributos.
- Entidades sin declaraciones de clave.
- Ciclos en las definiciones de jerarquías de generalización.
- Huérfanos (entidades desconectadas).

Dependiendo del grado de detalle con que se especifiquen las transacciones, la herramienta puede detectar atributos a los que hacen referencia las

transacciones pero que no están declarados en ninguna entidad, y transacciones que hacen referencia a entidades no enlazadas adecuadamente por interrelaciones. Por ejemplo, una transacción común en la base de datos de un hospital es asignar una habitación de exploración para un paciente con cita previa. La transacción hace referencia a atributos como NUM_EXAMEN y NUM_HABITACION, pero la herramienta podría ver que no se especificó interrelación alguna entre las entidades NUM_EXAMEN y NUM_HABITACION, donde ocurren estos atributos.

Un analizador de esquemas puede verificar también la compleción mutua de los esquemas de datos y funcionales, comprobando que los almacenes de datos dentro de los DFD estén representados en el esquema conceptual y que las operaciones de manipulación de datos (indicadas en los esquemas de navegación) pertenezcan a algún proceso.

15.4.6. Sintetizador de esquemas ER

Si bien no se sugiere este enfoque, es factible empezar el proceso de diseño conceptual de bases de datos especificando qué dependencias funcionales (DF) se deben cumplir. Como se expuso en el apartado 6.5, las DF definen las restricciones entre los atributos respecto a si deben o no ser únicos los valores que adopten. Por ejemplo, el atributo PRECIO puede depender funcionalmente del atributo NUM_ARTICULO. Dado un NUM_ARTICULO, digamos 105, el valor de PRECIO para ese artículo es único, digamos 1,29; no pueden existir dos valores de PRECIO para el mismo NUM_ARTICULO.

El diseñador puede especificar las DF con base en los requerimientos en lenguaje natural, en formularios o en formatos de registros existentes. Para empezar con las DF pero luego continuar con la descripción y refinamiento del esquema conceptual en el modelo ER, resulta muy útil una herramienta que sintetice un esquema ER a partir de las DF.¹ Tal herramienta produce, en primer lugar, un esquema relacional mediante la normalización (véase el apartado 6.5). Convierte cada relación en una entidad, que puede tener identificadas una o más claves aspirantes (candidatas). Al concatenar los nombres de los atributos de una de estas claves, la herramienta genera un nombre para cada entidad. Luego, con base en la comparación de los nombres de los atributos, postula interrelaciones entre las entidades.² Las interrelaciones se dejan sin nombre. La herramienta termina solicitando un algoritmo para la disposición automática del diagrama y mostrando al diseñador el esquema ER resultante.³

En la práctica, esta herramienta heurística produce un esquema ER inicial bastante aceptable. Desde luego, el diseñador necesita perfeccionar los nombres de las entidades generadas por la herramienta, y debe nombrar las interrelaciones. Además, puede encontrar interrelaciones falsas y borrarlas. Por ejemplo, la herramienta podría haber postulado erróneamente una interrelación entre las entidades USUARIO y FABRICA porque ambas tienen un atributo DIRECCION.

15.5. Herramientas básicas para el diseño lógico

Como es evidente en capítulos anteriores de este libro y en la secuencia de herramientas de la arquitectura de referencia de la figura 15.2, las herramientas de diseño lógico se aplican a los resultados de las herramientas de diseño conceptual.

15.5.1. Evaluador del acceso a registros lógicos

Los evaluadores del acceso a registros lógicos ayudan a automatizar parcialmente los procesos descritos en el capítulo 11 (véase también Teorey, Yang y Fry [1980]). Como entrada, requieren una **definición de mezcla de transacciones** (una combinación de transacciones ponderadas por frecuencia). Las transacciones a su vez se definen en términos de especificaciones del lenguaje de acceso, recuentos de selección de casos, y esquemas de navegación, o mediante otros tipos diversos de secuencias descriptivas de operaciones de lectura/escritura sobre entidades e interrelaciones. Las salidas son recuentos de **accesos a registros lógicos (ARL)** organizados por entidad y transacción, y también para toda la combinación.

Si la herramienta sólo produce los ARL (como sucede con frecuencia), está en manos del *diseñador* cambiar el diseño de base de datos (y las especificaciones de transacciones que lo acompañan) y ejecutar repetidamente la herramienta para tratar de determinar: 1) si los datos derivados deben mantenerse en el esquema, y dónde; 2) cómo deben modelarse las jerarquías de generalización; y 3) si algunas entidades deben dividirse o fusionarse. Las herramientas más avanzadas pueden encargarse de la iteración necesaria, usando los recuentos de ARL resultantes para responder automáticamente estas preguntas y recomendar un esquema lógico óptimo que minimice los ARL.

15.5.2. Traductor ER a relacional

Las herramientas para la transformación ER a relacional trabajan primero dentro del modelo ER, creando un esquema ER con estilo relacional que sea equivalente al esquema de entrada, y después traduciéndolo de manera bastante directa al esquema relacional correspondiente. Para obtener un esquema ER con estilo relacional, la herramienta sigue estos pasos (en el apartado 12.2 se dan más detalles al respecto):

1. *Elimina los atributos compuestos y polivalentes del esquema.*
2. *Asegura que cada entidad tenga declarada una clave primaria* copiando los atributos de clave de una entidad a otra. En el capítulo 12 se llamó a este proceso *eliminación de identificadores externos*.
3. *Asegura que cada interrelación se base en la presencia de valores de atributos iguales* en las entidades implicadas, cosa que hace, una vez más, copiando

los atributos de clave de una entidad a otra. Esto permitirá considerar después estos atributos como pares clave/clave ajena en el modelo relacional. Por ejemplo, si existiera una interrelación de uno a muchos ASESORA entre los profesores y los estudiantes a quienes asesoran, pero ningún atributo en la entidad ESTUDIANTE para apoyar la interrelación, la herramienta copiaría la clave primaria de PROFESOR (digamos, el par APELLIDO y NOMBRE) en ESTUDIANTE. Una buena herramienta introduciría, asimismo, un comentario en esos campos para indicar que representan la relación ASESORA, lo que ayudará al diseñador a cambiarles el nombre después (manualmente) a APELLIDO_ASESOR y NOMBRE_ASESOR (o nombres informativos similares).

4. *Convierte las relaciones de muchos a muchos en una entidad de enlace y dos interrelaciones de conexión.*

Una vez que la herramienta haya creado un esquema ER de estilo relacional mediante estos cuatro pasos, el esquema se podrá traducir directa y automáticamente al modelo relacional *haciendo de cada entidad una relación y descartando todas las interrelaciones*. Las principales diferencias entre las distintas herramientas de transformación automatizada ER a relacional son: 1) qué tanto procesamiento realizan antes de lograr la transición al modelo relacional, y 2) la atención que ponen para la transformación cuidadosa de las restricciones (Rosenthal y Reiner, 1987).

Antes de popularizarse el modelo relacional, algunas organizaciones construyeron sus propias herramientas para el diseño de bases de datos de redes y jerárquicas. Por ejemplo, IBM definió varias herramientas de diseño para producir esquemas IMS, incluido DBDA (*Database Design Aid*, ayuda para el diseño de bases de datos).

15.6. Herramientas actuales de diseño de bases de datos orientadas a la investigación

En los siguientes apartados se describen brevemente cinco herramientas de investigación y juegos de herramientas: DDEW, SECSI, Gambit, SIT (*Schema Integration Tool*, herramienta de integración de esquemas) y VCS (*View Creation System*, sistema de creación de vistas). Estos sistemas ilustran diversas ideas, interfaces y enfoques innovadores para apoyar el diseño de bases de datos. Estos y otros prototipos de investigación han influido y contribuido a la funcionalidad de varias herramientas comerciales actuales (de hecho, SECSI está a punto de ser dado a conocer como producto por INSOSYS). Otros sistemas de investigación acerca de los cuales no está de más leer incluyen COMIC (Kangassalo, 1989); TAXIS (Mylopoulos et al., 1984); RIDL* (De Troyer, Meersman y Verlinden, 1988; De Troyer, 1989); DATAID (Albano et al., 1985) y los sistemas descritos en Choobineh et al. (1988), Dogac et al. (1989) y Ceri (1983). RIDL* es comercializado actualmente por IntelliBase en Bélgica.

15.6.1. Mesa de trabajo de diseño y evaluación de bases de datos (DDEW)

DDEW (*Database Design and Evaluation Workbench*) fue construido por la división de investigación de Computer Corporation of America² (Reiner et al., 1986). Apoya el diseño de bases de datos desde el análisis de los requerimientos hasta el diseño físico; utiliza un modelo ER extendido (ER+) para el diseño conceptual, y permite elegir un modelo relacional, de redes o jerárquico para el diseño lógico. El marco de DDEW para el refinamiento progresivo y la iteración permite generar, presentar, manipular, analizar y transformar los diseños. La interfaz con el usuario está integrada semánticamente a la totalidad del proceso de diseño; el sistema representa y gestiona de forma explícita las derivaciones del diseño, los niveles de abstracción, los diseños alternativos y los subconjuntos del diseño.

DDEW se aplicó por primera vez en una estación de trabajo Jupiter y luego se transportó a una Sun. El sistema se apoya en múltiples ventanas de la pantalla y un **árbol de diseño** para presentar historias de diseño y niveles jerárquicos del detalle del diseño, y para ayudar al usuario a navegar a través de ellos. La pantalla de la figura 15.4 muestra un árbol de diseño y varias ventanas asociadas con uno de los diseños que contiene. Para presentar y ayudar a clarificar las estructuras del diseño, DDEW utiliza colores contrastantes e iconos gráficos, incluida una representación gráfica de las cardinalidades de las interrelaciones que reemplaza los pares numéricos (mín, máx) en la pantalla [se adoptó una pequeña variante de esta notación gráfica en Teorey (1990)]. Los diagramas de diseño grandes son apoyados por diversos mecanismos: desplazamientos, realce de **grupos afines** y una **ayuda de navegación** en miniatura para visualizar un diagrama de diseño entero y moverse rápidamente dentro de él.

Las herramientas de DDEW se encargan de la edición gráfica, la organización automática de diagramas ER, la generación de un esquema ER inicial a partir de dependencias funcionales, la integración de vistas, la normalización, la traducción de modelos de datos, la verificación de la integridad de los esquemas, la evaluación de los accesos a registros lógicos y físicos para una determinada combinación de transacciones, la selección de índices, y la carga y generación del esquema. Gestiona un conjunto bastante rico de restricciones de entidades y atributos, incluidos los tipos de datos, los atributos no nulos, las claves, las dependencias funcionales y de inclusión. También hay varias restricciones sobre las interrelaciones.

15.6.2. Sistema experto para el diseño de sistemas de información (SECSI)

SECSI fue un proyecto de investigación en el Laboratorio MASI del Instituto de Programación de la Universidad de París (Bouzeghoub et al., 1985). Más bien orientado hacia el diseño lógico de bases de datos, SECSI se construyó sobre el

² Ahora conocido como Xerox Advanced Information Systems (Sistemas Avanzados de Información Xerox).

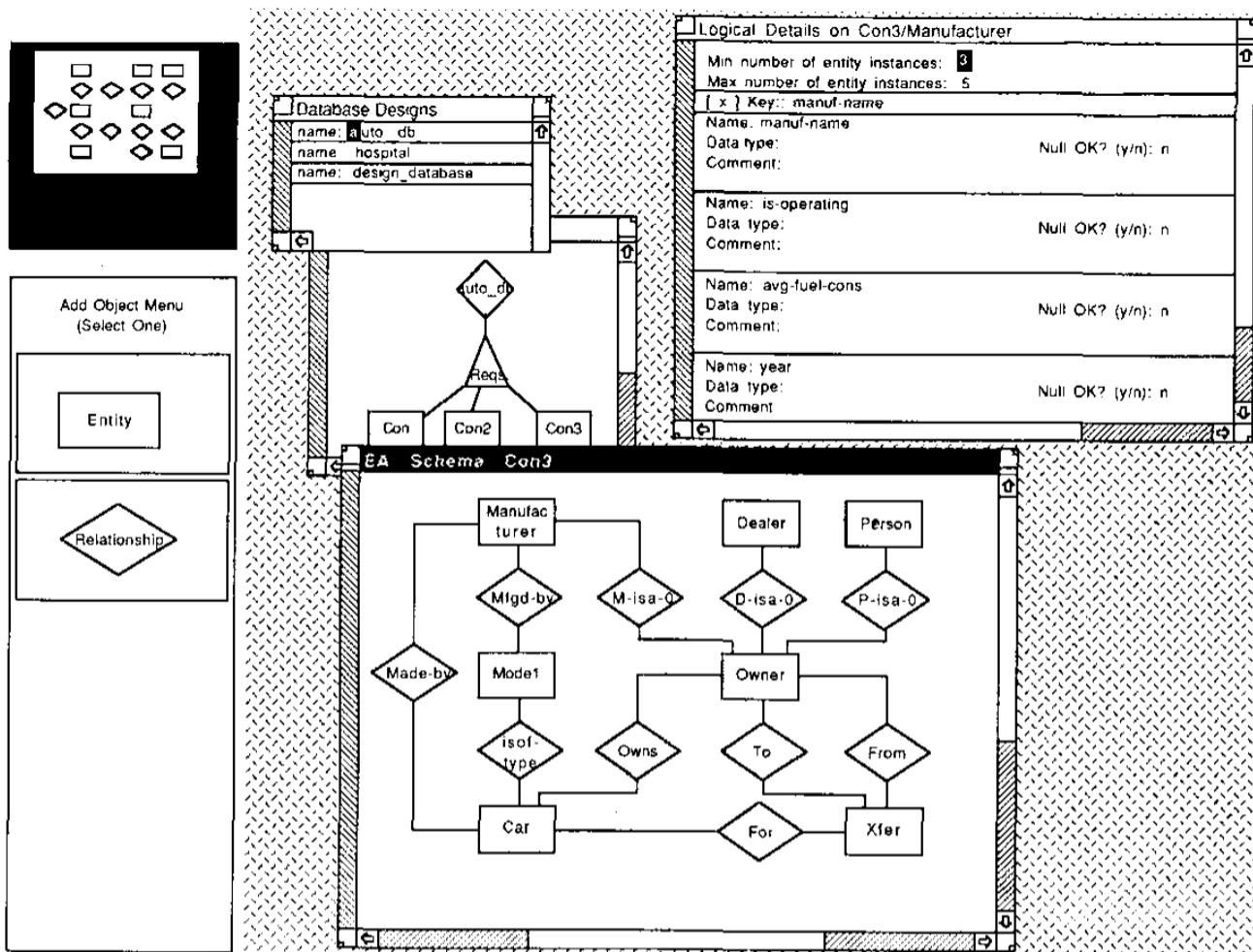


Figura 15.4. Interfaz con el usuario de DDEW.

DBMS relacional SABRE y se propuso como entorno metodológico interactivo y abierto. SECSI ya se ha transportado a la estación de trabajo Sun bajo Unix y funciona sobre el DBMS SABRINA de INFOSYS.

Las herramientas y los algoritmos de SECSI se encargan de la normalización y la producción de un esquema relacional, una vez que las entidades, interrelaciones, dependencias funcionales, cardinalidades de las interrelaciones y otra información ha sido solicitada al diseñador. Permite entradas en lenguaje natural limitado o bien gráficas, y produce tanto relaciones base como relaciones virtuales o vistas. Existen versiones en inglés y francés. Una característica poco usual es que SECSI puede interpretar cambios graduales a las entradas en lenguaje natural. El sistema gestiona dominios y restricciones referenciales y de inclusión. Internamente, SECSI representa los datos como una red semántica y (en la versión para Sun) permite al usuario personalizar los símbolos gráficos usados para presentar las entidades, interrelaciones, etcétera.

SECSI fue desarrollado como sistema experto para el diseño de bases de datos. Dentro de su base de reglas (más de 800) las hay que obligan a mantener la coherencia y que transforman la estructura de los esquemas, así como conocimientos generales sobre los conceptos ER y relacionales. Existe, asimismo, una jerarquía de metarreglas que controla la secuencia de los pasos del diseño, las reglas que se aplican en cada paso, y los hechos sobre los que operan. Por ejemplo, una de esas metarreglas describe una estrategia de búsqueda en profundidad («primero hacia abajo») para localizar y suprimir las jerarquías de generalización durante el diseño lógico. Las reglas están en PROLOG, en la forma «si condición, entonces acción». Con frecuencia, las condiciones se basan en las interrelaciones entre pares de objetos; por ejemplo, «si X es una generalización de X1 y A es un atributo de X, entonces A es un atributo de X1». Una interfaz interactiva adicional, llamada *función de aprendizaje (Learn)*, permite a un experto en diseño de base de datos modificar o añadir reglas de diseño. Por su orientación de sistema experto, SECSI acepta especificaciones incompletas, justifica y explica sus resultados y permite retroceder a cualquier paso del diseño para hacer modificaciones o explicarlo. La figura 15.5 muestra una explicación típica (con ediciones muy pequeñas para mejorar la traducción de francés a español).

15.6.3. GAMBIT

Desarrollado en el Instituto Federal Suizo de Tecnología (ETH Zurich), GAMBIT se basa en un modelo ER/relacional extendido (Braegger et al., 1985). Está asociado con el sistema LIDAS de bases de datos para el computador personal Lilith. GAMBIT ayuda al diseñador a delinejar y describir estructuras de datos y las transacciones de actualización asociadas. Las restricciones se formulan usando el lenguaje de programación de base de datos Modula/R, que se basa en el cálculo de predicados de primer orden.

El diseñador puede alternar entre dos vistas de diseños intermedios: una muestra un diagrama de bloques global con todos los conjuntos de entidades e interrelaciones que se han definido; la otra muestra una especificación textual de un conjunto de entidades con todos sus detalles (claves, atributos y dominios). GAMBIT ayuda al diseñador a transformar cada interrelación m:n en una entidad de enlace y dos interrelaciones incidentes, y también ayuda a eliminar estructuras de datos recursivas. La idea de la normalización del esquema se extiende para asegurarse de que los atributos aparezcan sólo en un punto en las jerarquías de generalización.

Una vez definidas las estructuras de datos (al menos parcialmente), el usuario se centra en las restricciones, especificando en qué objetos están, las operaciones (por ejemplo, inserción, borrado y modificación de objetos) después de las cuales se deben verificar, y el modo de proceder si son violadas. El sistema incluye un modelado explícito de la semántica de propagación de la inserción y el borrado, incluidas prohibiciones sobre la propagación. Por ejemplo, un dise-

- Usuario: ¿Por qué es persona una relación virtual?
 - SECSI
1. Se introdujo PERSONA como entidad, con los atributos NSS, NOMBRE, y como supertipo de ESTUDIANTE, PROFESOR, ASISTENTE
 2. La estructura que sigue (a la que pertenece este objeto) se ha transformado así:

Sabiendo que: existen al menos 3 atributos en los subtipos: NUMERO, TEL Y DIRECCION; y los subtipos están implicados en 2 asociaciones: MATRICULADO y RESPONSABLE; y los subtipos ASISTENTE y PROFESOR tienen otro supertipo: EMPLEADO; he realizado esta transformación:

El supertipo PERSONA se ha suprimido; los subtipos se convierten en relaciones base. PERSONA es ahora una relación virtual y se expresa su derivación en los resultados.

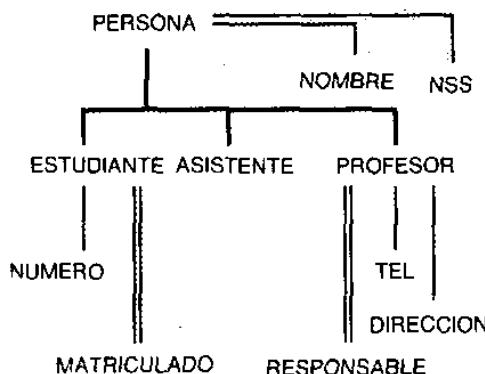


Figura 15.5. Explicación de SECSI de una decisión de diseño.

ñador puede establecer que si se añade un curso nuevo a una base de datos de una universidad, debe existir un profesor que «participe» en él (es decir, que lo imparte); pero si este profesor no existe, no puede ser añadido en la misma transacción. Esta restricción se define mediante la interfaz gráfica mostrada en la figura 15.6.

Una vez establecidas las restricciones, GAMBIT genera automáticamente (como módulos de base de datos en MODULA/R) las transacciones adecuadas de inserción, borrado y modificación que obligan al cumplimiento de la semántica de propagación. Luego, el recurso de prototipos del sistema permite al diseñador probar el diseño y mejorar de forma interactiva el esquema conceptual y las transacciones correspondientes.

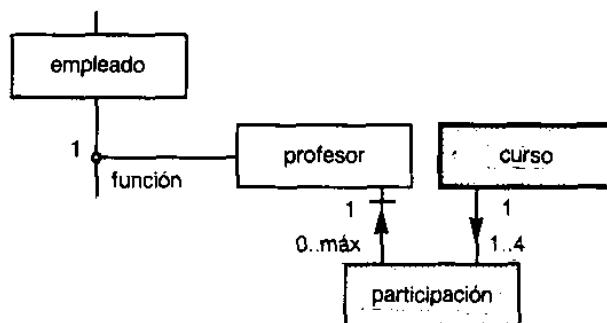


Figura 15.6. Propagación de la transacción «insertar curso» en GAMBIT.

15.6.4. Herramienta para la integración de esquemas (SIT)

La herramienta para la integración de esquemas (SIT, *Schema Integration Tool*) fue un proyecto conjunto de Honeywell y la University of Florida, en una estación de trabajo Apollo (Navathe, Elmasri y Larson, 1986; Sheth et al., 1988; Elmasri et al., 1986). Esta herramienta reúne vistas en una variante del modelo ER llamada modelo de *entidades-categorías-interrelaciones (ECI)* (Elmasri et al., 1985). Las correspondencias de atributos entre las vistas se reúnen desde un principio en forma de interrelaciones de dominio entre los atributos (por ejemplo, «concordancia exacta», «A contenida en B» o «A disociada de B»). El sistema verifica la coherencia de la información suministrada por el diseñador y en seguida guía a este último a través de los pasos de detección y resolución de conflictos, utilizando procedimientos heurísticos basados en la semejanza de los atributos.

El diseñador puede decidir si dos entidades o interrelaciones deben fusionarse o no, y si ciertas abstracciones son significativas (por ejemplo, si ESTUDIANTE y EMPLEADO deben generalizarse para dar una entidad superior, digamos PERSONA, que contenga sus atributos comunes). La herramienta hace hincapié en un procedimiento de integración n-ario (en oposición al enfoque de pares de DDEW). Se integran todas las n vistas una vez que las correspondencias entre atributos y entre entidades se aprueban y confirman en consulta con el diseñador. Se usa un solo algoritmo para realizar la integración tanto de entidades como de interrelaciones.

15.6.5. Sistema para la creación de vistas (VCS)

El sistema experto VCS (*View Creation System*) para el diseño de bases de datos se desarrolló en la University of British Columbia y fue implantado en PROLOG en un Amdahl 5850 (Storey y Goldstein, 1988). Más tarde se cambió a un entorno de microcomputador. VCS establece un diálogo con el diseñador sobre los requerimientos de información de una aplicación, desarrolla el esquema ER co-

rrespondiente y luego convierte el esquema en relaciones en la cuarta forma normal (4NF)³. El sistema se basa en la premisa de que existe una escasez de diseñadores de bases de datos con experiencia para reunir información de los usuarios finales y que, por tanto, tiene sentido automatizar la tarea inicial de especificación de vistas.

La metodología para la creación de vistas se representa como un conjunto de reglas (alrededor de 500) reunidas dentro de una base de conocimientos. Algunas son **reglas de procedimiento**:

Primero: Identificar las entidades, sus atributos y claves aspirantes (candidatas).

Luego: Determinar las interrelaciones, atributos de interrelaciones y razones de correspondencia.

Luego: Detectar y resolver las ambigüedades, redundancias e incongruencias.

Algunas son **reglas de producción**:

Si: Una interrelación es de la forma A ES_UN B

Entonces: Representar la interrelación incorporando la clave de la entidad B como clave ajena de A.

Mediante un diálogo interactivo con el usuario (Fig. 15.7), el sistema trabaja hacia fuera a partir de las entidades básicas. Detecta atributos polivalentes y repetidos, y los separa como entidades individuales. Inserta interrelaciones adicionales cuando están implicadas por atributos que hacen referencia a otras entidades; junta o genera combinaciones únicas de campos clave, descubre las cardinalidades de las interrelaciones, y así sucesivamente. Algunos nombres de interrelaciones (por ejemplo A TIENE B) son inherentemente ambiguos; VCS formula preguntas para averiguar si B es un *componente de A*, es un *caso de ella*, está *asociada a ella* o es *poseída por ella*. El sistema resuelve, asimismo, las incongruencias que surgen al definir interrelaciones jerárquicas o del uso accidental de un sinónimo. La normalización y definición de las interrelaciones son relativamente claras y no requieren interacción del usuario con el sistema.

15.7. Herramientas actuales de diseño de bases de datos comerciales

Los sistemas detallados en este apartado y el próximo —MastER Plus, ERwin, Bachman Re-Engineering Product Set, Excelerator, POSE, Information Engineering Workbench y AD/Cycle Repository— son ejemplos de las docenas de

³ Si el lector desea una exposición de la cuarta forma normal, puede consultar el capítulo 14 de *Fundamentals of Database Systems*, por R. Elmasri y S.B. Navathe (Benjamin/Cummings, 1989).

Sistema: Dime qué cosas son de interés para ti en una «circulación_de_biblioteca».

Me referiré a estas «cosas» como «entidades».

Asimismo, quiero que me des los «atributos» de estas «entidades», es decir, las propiedades o características de interés para ti.

[El sistema da un ejemplo de entidades y atributos a partir de una base de datos de reservas de una línea aérea.]

Toma nota de que puedes usar una «palabra» para cada atributo, pero la «palabra» puede consistir en varias palabras conectadas por caracteres de subrayado.

Si introduces algo y después decides que es erróneo, lo puedes cambiar luego.

Ahora, por favor, introduce el nombre de una entidad en su forma SINGULAR.

Usuario: prestatario.

S: ¿Cuáles son los atributos de «prestashopario»?

Introdúcelos separados sólo por un espacio y con un punto después del último.

O bien, introduce «fin», si realmente no los conoces.

U: Número nombre dirección teléfono libro1 libro2 libro3

S: ¿Son «libro1» «libro2» «libro3» diferentes ocurrencias de «libro»? «sí»/«no»/«¿por qué?»

U: ¿Por qué?

S: Parece que «libro» «libro2» «libro3» se refieren todos a la misma «cosa» -- esto es, «libro» Una «cosa» debe ser una entidad.

Ahora, ¿son «libro1» «libro2» «libro3» todos ocurrencias de «libro»? «sí»/«no»/«¿por qué?»

U: Sí

S: Entonces, haremos a «libro» una entidad cuando hayamos terminado con ésta.

Figura 15.7. Ejemplo de diálogo de un sistema vcs con un usuario que está diseñando una vista para una base de datos de circulación de una biblioteca.

sistemas comerciales disponibles. Nuestro intento no es persuadir al lector de que adquiera uno u otro de ellos; realmente somos neutrales al respecto. Más bien, el objetivo es presentar sistemas típicos comercialmente disponibles. (En el apartado 15.9 hay una visión de conjunto tabulada de muchos de los sistemas comerciales actuales).

15.7.1. MastER Plus

Desarrollado inicialmente como sistema de investigación italiano, MastER Plus fue comercializado en PC por Gestione Sistemi per l'Informatica (GESI) y es distribuido en Norteamérica por Infodyne International, Inc. El sistema apoya una metodología similar a la descrita en este libro. En la primera fase de diseño, se captan los requerimientos del usuario dividiendo las aplicaciones grandes y complicadas en varias aplicaciones locales. Los subesquemas conceptuales que corresponden a las aplicaciones locales consisten en muchos tipos de componentes: entidades, interrelaciones, jerarquías, atributos, identificadores, nodos de red, procesos, almacenes de datos, interfaces, usuarios y sucesos. Toda esta información está almacenada en una base de datos de proyectos, un depósito central que se usa en las siguientes fases. Una faceta interesante de MastER Plus es que el metaesquema de la propia base de datos de proyectos (Fig. 15.8) puede

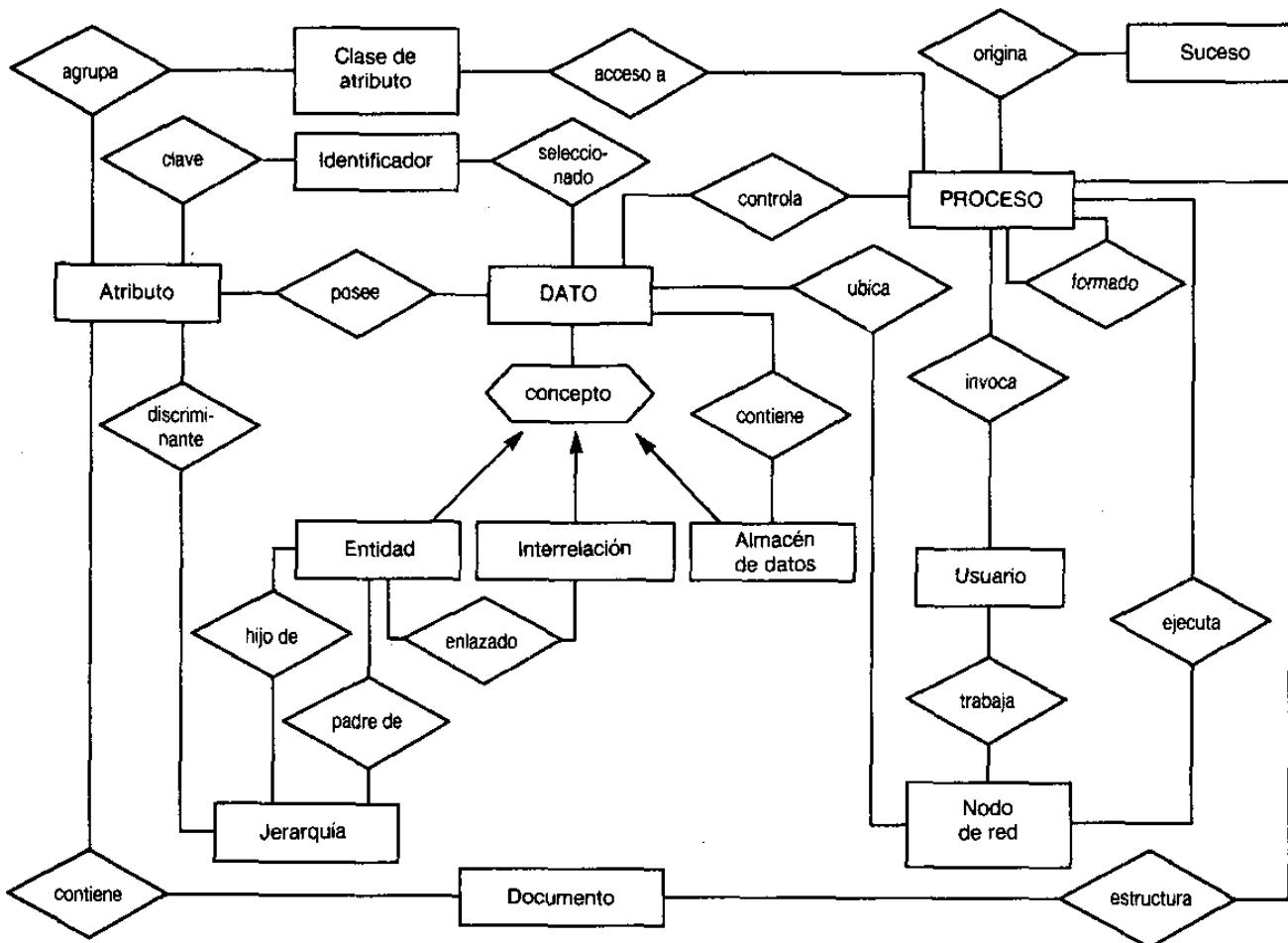


Figura 15.8. Metaesquema ER de MastER para la base de datos de diseño conceptual.

examinarse de la misma forma que los esquemas de las aplicaciones de bases de datos que el sistema ayuda a desarrollar.

En la segunda fase de diseño, el diseñador integra los subesquemas conceptuales en un esquema conceptual único. La integración se hace por pares, empezando idealmente con los esquemas a los que se ha dado menor peso y finalizando con los de más peso. La herramienta detecta conflictos (por ejemplo, incongruencias de tipo entre objetos con igual nombre) entre los dos esquemas y propone soluciones al diseñador. El diseñador corrige los esquemas según la necesidad. La herramienta los fusiona y analiza el borrador del esquema integrado para descubrir redundancias y simplificar la representación.

En la tercera fase de diseño, el esquema conceptual integrado se convierte automáticamente en un esquema relacional (Catarci y Ferrara, 1988). Se puede elegir entre dos diferentes algoritmos de conversión. Uno produce un esquema normalizado estándar; el otro produce un esquema relacional optimizado, con índices y utilización del almacenamiento afinados a la aplicación objetivo.

En la cuarta fase de diseño, el diseñador puede optimizar aún más el esquema relacional en el nivel físico si así lo desea. El sistema puede entonces generar automáticamente DDL para varios DBMS (ADABAS, dBase III, ORACLE, IBM S/38 y AS/400, db2, Informix y RBase).

Para finalizar, MastER Plus puede generar automáticamente un prototipo gráfico de la aplicación basado en el esquema y en información relacionada con el diseño. Esto incluye máscaras de pantalla, menús, programas y grupos de usuarios con diferentes contraseñas y conjuntos de funciones. El prototipo resuelve consultas, actualizaciones, borrado, presentación de esquemas y (opcionalmente) ayuda en línea. Los programas generados son, en realidad, programas dBase III compilados con Clipper; pueden refinarse y enlazarse con el módulo DrivER del sistema para desarrollar las aplicaciones funcionales conectadas en red.

15.7.2. ERwin

ERwin es una herramienta de diseño basada en Windows, producida por Logic Works Inc., que ofrece una interfaz en su mayor parte gráfica para la variante IDEF1-X de ER (Fig. 15.9). Los datos gráficos del diagrama se introducen mediante el ratón y la caja de herramientas (una galería o paleta de elementos de diagrama) y se puede editar las trayectorias de las interrelaciones. Los datos textuales, incluidas notas y definiciones, se introducen a través del teclado con editores que facilitan el diálogo. Las claves, claves alternas, claves ajenas y restricciones de integridad referencial se introducen de forma explícita. El diseñador puede, asimismo, introducir muestras de casos y de consultas.

ERwin ofrece un alto grado de apoyo automático de modelado. Las claves ajenas se propagan automáticamente a registros de enlace y subtipos. La herramienta cambia las entidades de independientes a dependientes y viceversa cuando resulta apropiado. Si se borra una entidad o interrelación, ERwin borra todas las claves ajenas que hacían referencia a ella. Los atributos dentro de una entidad se unifican si provienen del mismo padre, pero el diseñador puede anular la unificación especificando atributos de «nombramiento de papeles». La herramienta realiza la normalización, imprime diversos informes y genera esquemas SQL compatibles con DB2.

15.7.3. Conjunto de productos de retroingeniería Bachman

El conjunto de productos de retroingeniería Bachman (*Bachman Re-Engineering Product Set*), de Bachman Information Systems, Inc., se centra en la retroingeniería a partir de esquemas existentes, con ayuda experta de una base de conocimientos integrada (Fig. 15.10) y de un entorno de trabajo gráfico flexible (Bachman, 1988). La herramienta BACHMAN/Data Analyst puede crear un esquema ER detallado automáticamente, en combinación con un producto acompañante que extrae información de una fuente de datos: IDMS, IMS, archivos

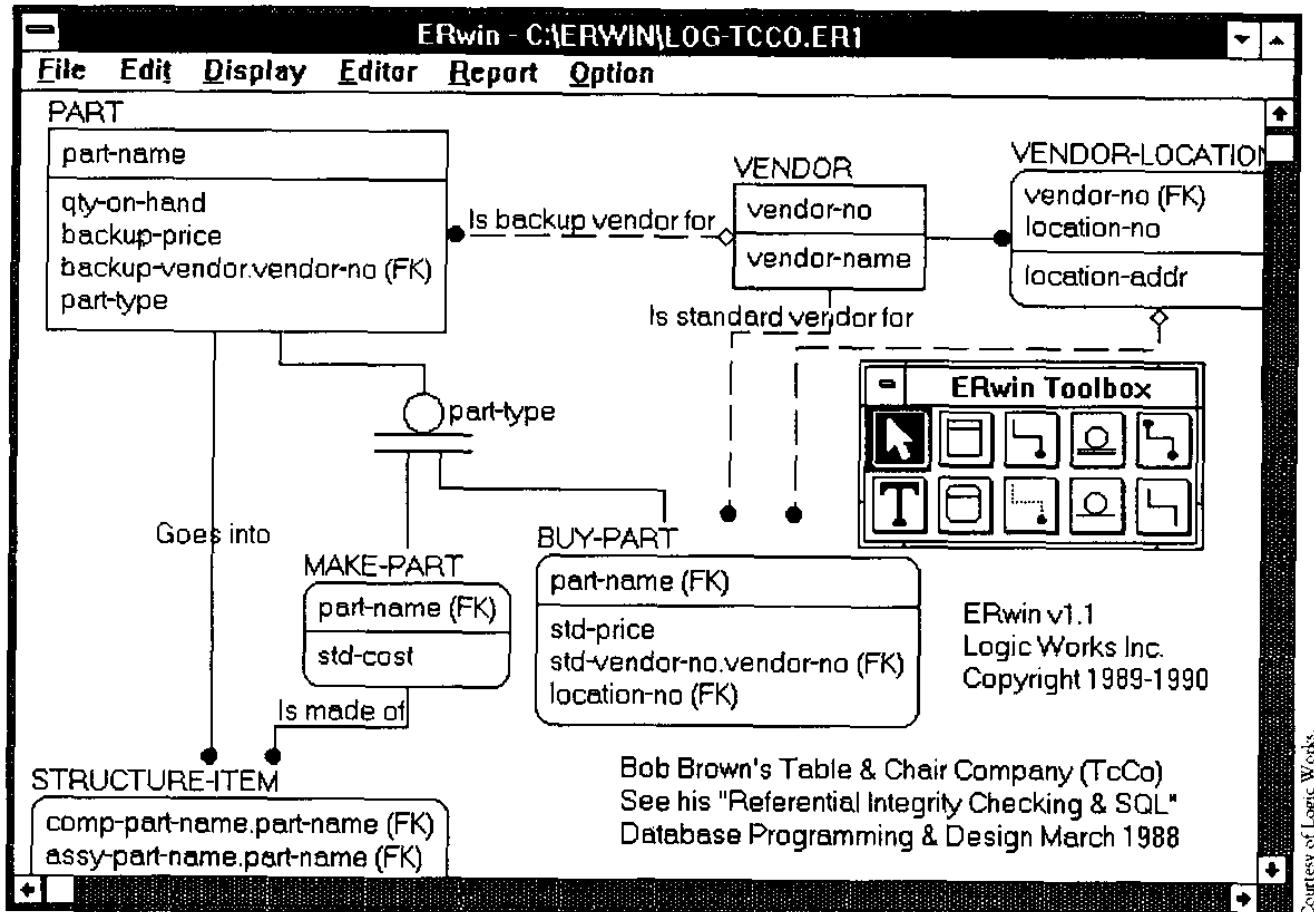
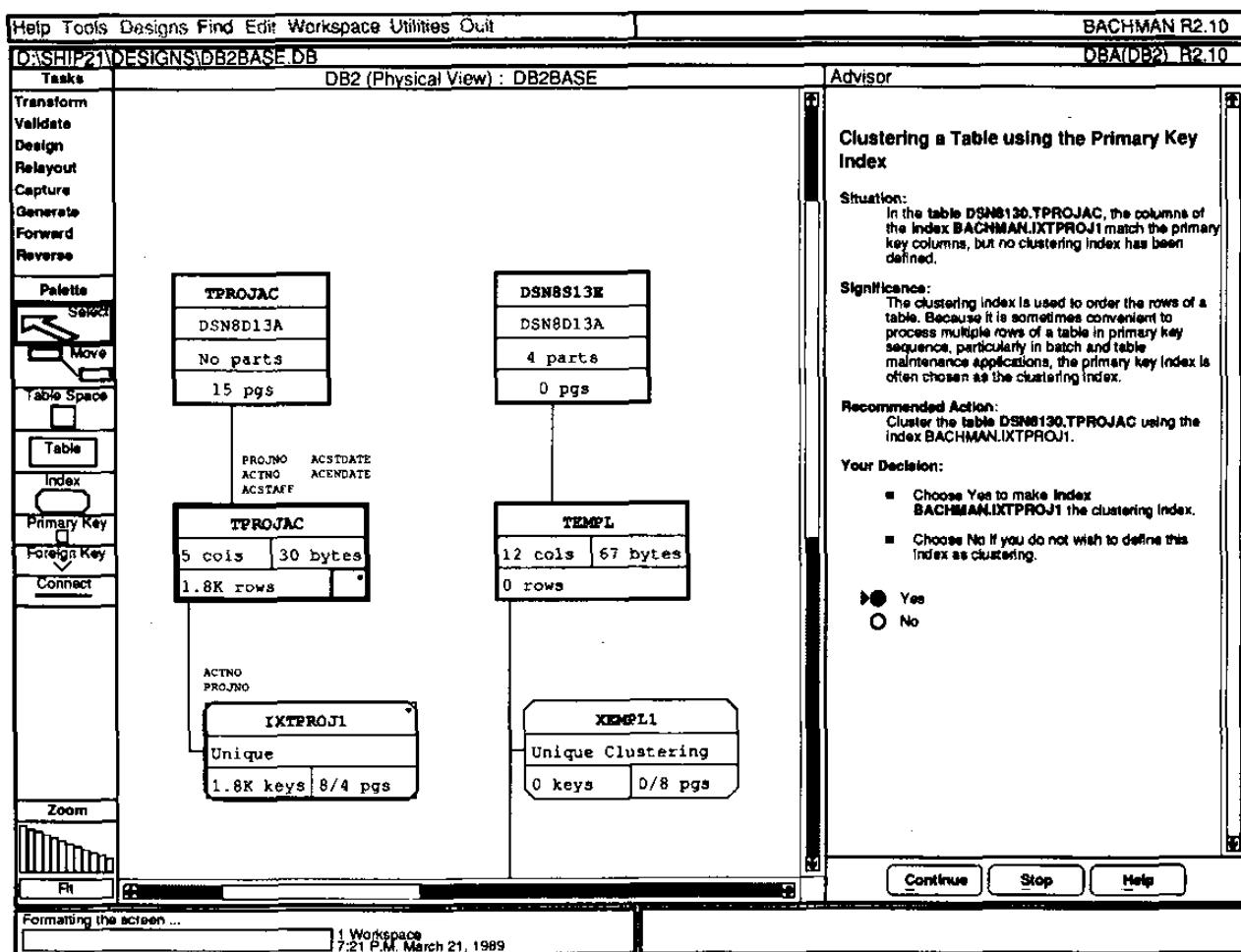


Figura 15.9. Diseño conceptual en ERwin usando el modelo IDEF1-x.

planos VSAM o DB2. Como alternativa, el diseñador puede construir un diagrama ER a partir de cero, usando el editor gráfico y los editores de texto asociados. El diseñador puede realizar desplazamientos, acercamientos y alejamiento dentro del diagrama.

La herramienta Data Analyst capta entidades, interrelaciones, claves primarias, jerarquías de tipos de entidades (incluida la herencia de propiedades) y estimaciones de volumen y crecimiento de entidades. Respecto a las interrelaciones, capta las cardinalidades, los volúmenes (mínimo, máximo y estimado), interrelaciones complejas con operadores booleanos como AND y OR, y definiciones de claves ajenaas derivadas. Data Analyst realiza la normalización y otra herramienta, BACHMAN/DBA, efectúa proingeniería, es decir, la producción de esquemas a nivel físico, para DB2 e IDMS. La herramienta DBA para DB2 puede calcular el espacio libre y ayuda a seleccionar índices (incluido el ordenamiento de las columnas en un índice multicolumnas); genera DDL incremental según el estado actual del catálogo DB2.

El diseñador puede elegir entre cinco niveles de asistencia experta integrada:



Courtesy of Bachman Information Systems, Inc.

Figura 15.10. BACHMAN/DBA ayuda al diseñador en la fase de diseño físico.

desde un nivel altamente didáctico, adecuado para el novato, hasta una versión moderada de asistencia más apropiada para un administrador de bases de datos experimentado. Las bases de reglas del producto comprenden: uso coherente de nombres, estimación de volúmenes, análisis de clave ajena de atributos, dominios y dimensiones, eliminación de artefactos de implantación; prácticas alternativas de modelado, análisis de jerarquías de tipos de entidades, validación de compleción del modelo ER; selección de índices, elección de la implantación, agrupamientos y ordenamiento de columnas.

15.8. Herramientas CASE comerciales actuales que permiten diseñar bases de datos

15.8.1. Excelerator

Excelerator de Index Technology integra diagramas y herramientas de diseño de bases de datos y de análisis de sistemas, y posee apreciables características de personalización (Index Technology Corp., 1987; Williams, 1988). Almacena entidades, atributos, objetos e interrelaciones en su propio diccionario de diseño local, XLDictionary, pero también es capaz de utilizar el Repository Manager de AD/Cycle de IBM (véase el apartado 15.8.4) y el entorno de desarrollo CASE VAX CDD/Plus de DEC.

Excelerator puede utilizar diagramas de flujos de datos Gane/Sarson así como Yourdon, diagramas de estructuras de Jackson, los gráficos de estructura de Constantine, diagramas de modelos de datos Bachman y diagramas ER tanto estilo Chen como estilo Merise (Quang, 1986). Si el diseñador lo cree oportuno, los objetos y flujos de cada gráfica se pueden expandir en gráficas, entidades o descripciones textuales más detalladas.

El sistema puede aplicar retroingeniería cargando definiciones de bases de datos en SQL existentes. Puede informar sobre datos redundantes y violaciones de la normalización, así como conflictos de acceso a elementos y tipos de registros con claves iguales. Permite al diseñador describir restricciones, vistas e índices de tablas con mucho detalle (Fig. 15.11). Por ejemplo, Excelerator capta la decisión *restringir/hacer-nula/propagar* para una restricción de integridad referencial. Al diseñar en DB2, Excelerator calcula automáticamente el espacio en pistas y cilindros a nivel de diseño físico.

Un producto acompañante flexible, Customizer, permite a los usuarios modificar Excelerator para apoyar las necesidades, técnicas y métodos específicos de una organización. La clave de esta claridad es que gran parte de la conducta de Excelerator se basa en dos archivos centrales: el diccionario del sistema y la biblioteca de formatos del sistema. Un usuario puede modificar estos archivos para añadir nuevos tipos de entidades y atributos, cambiar la estructura de menús o incluir nuevos tipos de gráficas con nuevos objetos gráficos. Por ejemplo, el XLDictionary puede ser personalizado para incluir nuevos tipos de entidades e interrelaciones que apoyen una metodología para el análisis estructurado de una organización específica.

15.8.2. Ingeniería de software orientada a las imágenes (POSE)

Producido por Computer System Advisors, POSE (*Picture Oriented Software Engineering*) ofrece una integración de múltiples tipos de diagramas y módulos de diseño y un conjunto de herramientas muy modular (Pace, 1989). Las nueve herramientas de ingeniería informática se pueden usar para el análisis y diseño

Table CUSTOMER											
Logical Name	CUSTOMER										
Definition	ORDER PLACING CLIENT										
Valid/Edit Proc.	CUST										
Short Name	CUST										
Table Prefix	DDC										
Element Name	#	N	D	P	1	2	3				
CUSTOMER NUMBER	1	N	N	N	K						
CUSTOMER NAME	2	N	N								
CUSTOMER STREET ADDRESS	3	N	N								
CUSTOMER CITY	4	N	N								
CUSTOMER STATE	5	N	N								
CUSTOMER COUNTRY	6	N	N								
CUSTOMER POSTAL CODE	7	N	N								
CUSTOMER TELEPHONE NUMBER	8	N	N								
MANAGER NUMBER	9	N	N	N	F						
CUSTOMER SINCE	10	N	N								
CUSTOMER PRIORITY CODE	11	N	N								
CUSTOMER LAST ORDER DATE	12	N	N	N							

Legend:

- N = Null
- D = Default
- P = Primary Key[K1-9]
- F = Foreign Key[F]
- 1 = Secondary Keys[KS1-9]
- 2 = Secondary Keys[KS1-9]
- 3 = Secondary Keys[KS1-9]

PgDn

Courtesy of Index Technology Corporation.

Figura 15.11. Detalles de una definición de tabla en Excelerator.

de sistemas tanto orientados a los datos como orientados a procesos. Tienen una interfaz gráfica común y se integran mediante el diccionario de datos POSE. Del lado del diseño de base de datos, la secuencia de herramientas es: diagramador de modelos de datos (DMD, *Data Model Diagrammer*), normalizador de modelos de datos (DMN, *Data Model Normalizer*), diseñador lógico de bases de datos (LDD, *Logical Database Designer*) y ayuda de bases de datos (DBA, *Database Aid*).

El DMD está orientado a ER, pero permite hacer diseño lógico. Posee un sólido apoyo de diagramas con acercamientos (zoom), desplazamiento en pantalla, delimitación elástica y presentación concurrente de vistas globales y detalladas. Como ayuda para diseños grandes, los modelos de datos se pueden descomponer en varios modelos individuales que luego pueden recombinarse. En seguida, el DMN utiliza entidades, atributos, asociaciones (esto es, interrelaciones) y dependencias funcionales captadas en el modelo de datos para producir un esquema ER normalizado. Define claves y claves ajenas donde se necesita y detecta y corrige claves redundantes y dependencias transitivas.

Las transacciones se muestran como *mapas de utilización de transacciones* (TUMS, *Transaction Usage Maps*) encima del diagrama ER en el LDD (Fig. 15.12).

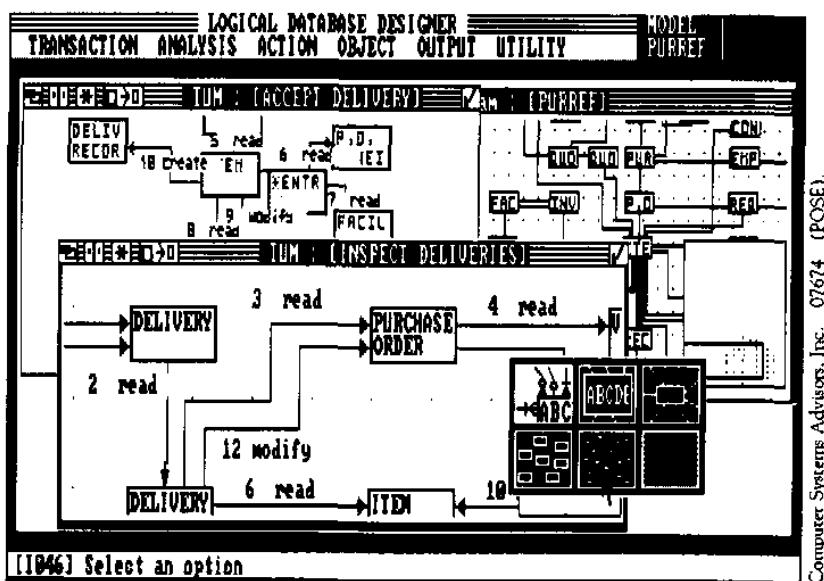


Figura 15.12. Mapas de utilización de transacciones (TUMS) en POSE.

Esta herramienta realiza las entidades a las que no tienen acceso las transacciones, las rutas de acceso a las que les faltan asociaciones, y las asociaciones que no se utilizan. Los TUMS captan los puntos de entrada de los datos, los volúmenes de utilización, y la secuencia de accesos y la lógica de las transacciones; a partir de esto, el LDD genera matrices de carga de utilización de rutas de acceso lógico. Por último, el módulo DBA crea DDL a nivel físico para DB2, SQL/DS, ADABAS, AS/400 y Focus.

POSE incluye también la mayoría de las herramientas CASE estándar, con algunas adiciones:

- Diagramador de descomposición (DCD, *Decomposition Diagrammer*) para representar las estructuras jerárquicas de organizaciones, sistemas, programas, archivos e informes.
- Diagramador de flujo de datos (DFD, *Data Flow Diagrammer*), con capacidades de niveles múltiples.
- Diagramador de gráficos de estructura (SCD, *Structure Chart Diagrammer*), para mostrar los componentes del sistema y sus interrelaciones.
- Diagramador de gráficos de acción (ACD, *Action Chart Diagrammer*), con el que los diseñadores de sistemas pueden desarrollar especificaciones de programas en un formato en inglés estructurado.
- Creador de prototipos de informes en pantalla (SRP, *Screen Report Prototyper*)
- Diagramador de matrices de planificación (PMD, *Planning Matrix Diagrammer*)

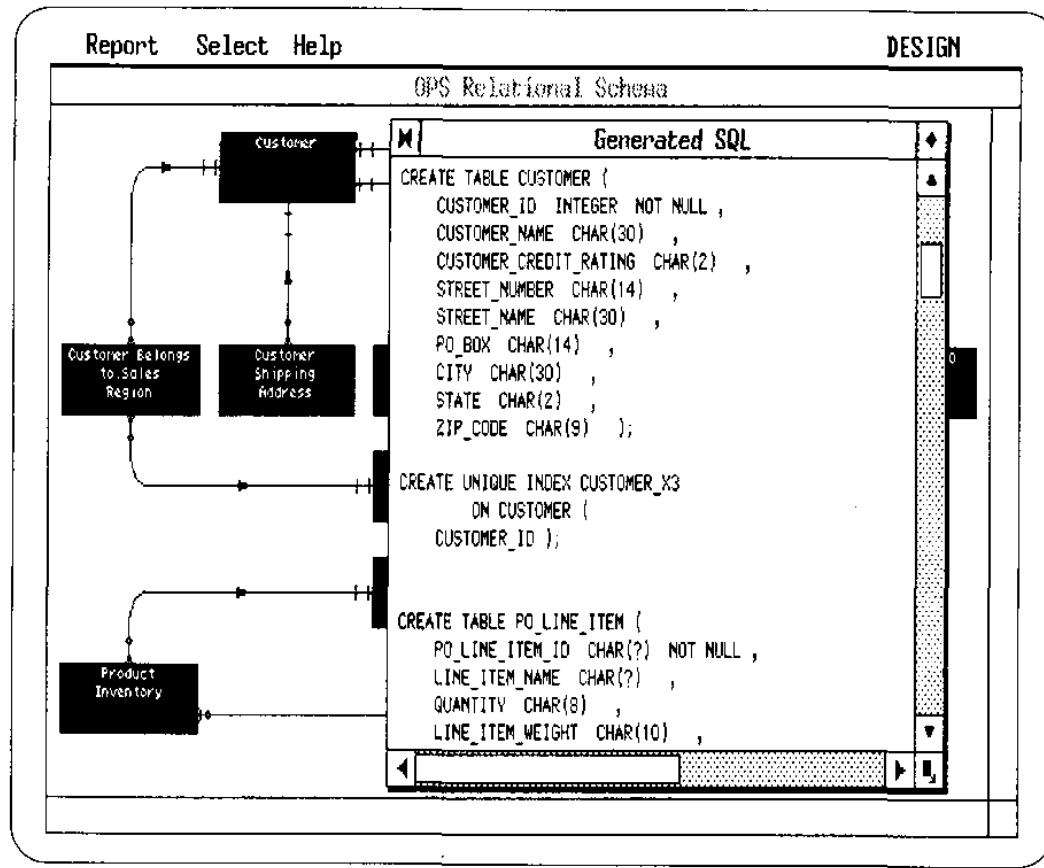
15.8.3. Mesa de trabajo de ingeniería de información (IEW)

El sistema IEW (*Information Engineering Workbench*) de Knowledge Ware, basado en PC y macrocomputadores, apoya múltiples tipos de diagramas e incluye herramientas de planificación, análisis y diseño. La integración entre diagramas se logra mediante un componente de enciclopedia; todas las herramientas tienen acceso a este depósito de información común. Una vez almacenado, el significado de un diagrama puede llamarse de nuevo y representarse automáticamente empleando cualquiera de los otros tipos de diagramas del sistema. Un componente basado en reglas, llamado *coordinador de conocimientos*, comprueba la validez del diagrama y de la información, y correlaciona la información entre los tipos de representación. Si dos diagramas contienen un elemento en común que se cambia en uno de ellos, el coordinador propaga el cambio al otro automáticamente. El coordinador de conocimientos contiene más de dos mil reglas.

Los diagramas IEW se codifican en color según el tipo. El diseñador abre subventanas en los diagramas ER para presentar y modificar las definiciones de las entidades y atributos, y ciertos detalles de las interrelaciones. Se muestran nombres en las dos direcciones de las interrelaciones, cosa que facilita la lectura de los diagramas ER. El diseñador puede enmascarar en un diagrama los elementos que distraen o resaltar la ruta de la información a través de varios diagramas.

Las herramientas IEW son capaces de obtener automáticamente un diagrama ER global a partir de diagramas ER individuales declarados para actividades, flujos de datos y almacenes de datos en un DFD, agentes externos y «áreas temáticas». Esto se realiza mediante la unión de los diagramas ER individuales y no abarca la resolución de conflictos. Además, el diseñador puede trazar diagramas ER que presenten los requerimientos de datos detallados de un diagrama de descomposición o de una actividad en un DFD. Para analizar el impacto de los cambios sobre las entidades, se puede solicitar un informe de referencias cruzadas sobre la aparición de las entidades en una vista DFD.

IEW es neutral respecto a la metodología, y apoya una gama de enfoques y técnicas que incluyen los de Yourdon, DeMarco, Gane/Sarson, James Martin y Arthur Young. Permite la retroingeniería de diseños de base de datos y tiene lazos estrechos con AD/Cycle Repository de IBM (véase el siguiente apartado). Dentro de la herramienta Database Designer/Relational (diseñador de base de datos/relacional), el diseñador puede solicitar DDL de SQL para los DBMS objetivo ORACLE o DB2 (Fig. 15.13). Mediante las capacidades del sistema abierto (importación/exportación) de IEW, los clientes de Knowledge Ware han desarrollado interfaces adicionales para varios DBMS, diccionarios de datos, y lenguajes de cuarta generación, incluidos Nomad, FOCUS, Model 204, Revelation, IBM Data Manager y RBase.



IEW R/Design Workstation Screen. © 1989 Knowledge Ware, Inc. Used with permission.

Figura 15.13. DDL de SQL generado por IEW.

15.8.4. Depósito de ciclos de desarrollo de aplicaciones

El depósito de ciclos de desarrollo de aplicaciones (*Application Development Cycle Repository*) es diferente de las tres herramientas CASE descritas anteriormente. El AD/Cycle Repository de IBM es una base de datos cuya intención es representar un punto único para compartir e intercambiar información sobre el desarrollo de aplicaciones entre las herramientas CASE (Fisher, 1990). Se le conoce como **panel posterior de software**, por analogía con un panel posterior de hardware, que sirve para conectar las diferentes tarjetas de circuitos en un computador. Controlado por DB2 en macrocomputadores, el depósito es parte de una familia de herramientas AD/Cycle que integran la estrategia de arquitectura de aplicación del sistema de IBM (SAA, *System Application Architecture*). AD/Cycle Repository es un elemento vital en el entorno de desarrollo necesario para apoyar un universo heterogéneo de aplicaciones que se ejecutan en una variedad de máquinas. Permite la definición de un modelo ER que abarca toda la empresa, y que se puede dividir en vistas específicas para cada aplicación.

El depósito ofrece un punto único de control para las definiciones de datos,

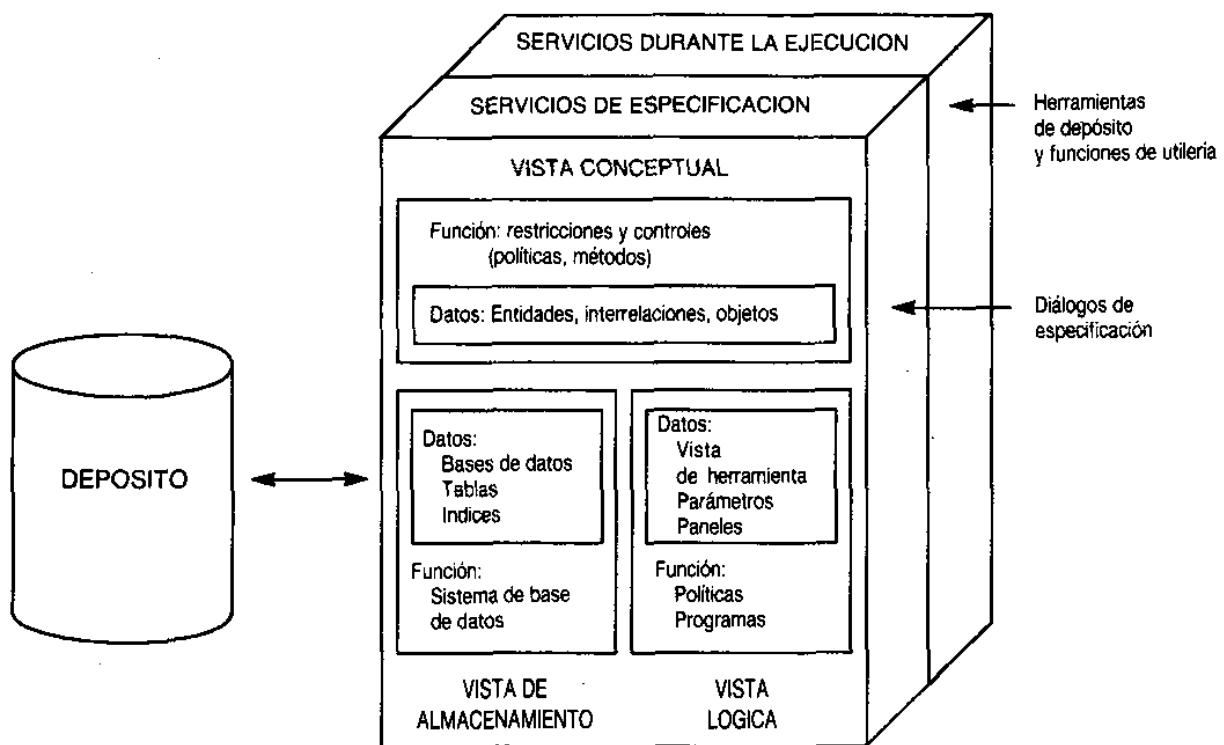


Figura 15.14. Modelo de función y datos del gestor del depósito [fuente: IBM].

objetos de aplicación y políticas (Fig. 15.14). Las **políticas** son una forma útil de extender el modelo del depósito para apoyar una metodología particular, y se dan en cuatro formas: seguridad, integridad, derivación y disparador. Aunque el depósito se encarga también del control de versiones, de acceso y de configuración, no contiene datos de producción, ni funciona como un DBMS de propósito general. Su papel como facilitador le distingue de los productos de catálogo y de diccionario de datos (Hazzah, diciembre de 1989).

Las herramientas CASE de diversos vendedores poseen interfaces para el depósito. Por ejemplo, en Excelerator de Index Technology, el depósito actúa como recurso de almacenamiento de metadatos de segundo nivel, basado en el anfitrión, complementando el propio XLDictionary de Excelerator. Vale la pena señalar que existen varios estándares y aspirantes a estándares de depósitos competitivos (Hazzah, agosto de 1989). Estos incluyen la norma de diccionario de recursos de información (*Information Resource Dictionary Standard*) de ANSI (Probandham et al., 1990) y CDD/Plus de DEC.

15.9. Resumen de herramientas comerciales

Las Tablas 15.1 y 15.2 ofrecen una muestra de los sistemas de diseño disponibles comercialmente. Se ha tratado de representar correctamente las caracterís-

Tabla 15.1. Sistemas orientados al diseño de bases de datos

Sistema de diseño	Compañía (teléfono)	Trabaja en	Modelos conc./fig.	Entrada del sistema	Normalización	Modelados de transacciones	Salidas del sistema	Comentarios sobre características del sistema
AD/VANCE Data Modeler	On-Line Software International (800-642-0177)	IBM MVS	ER/Rel	Diagramas ER	Si	No	DB2	Permite generalización y subconjuntos, validación extensa, acepta diagramas IEW como entrada
Blue/60	Interprogram (Netherlands 020-996121)	Mac	ER/Rel	Diagramas ER de Martin y detalles	Si	Si	Informes	Analisis de rutas de acceso
Colonel	Parker Shannon (617-787-7842)	PC	ER/Rel, Redes, Jer	Respuestas textuales a preguntas	Si	Si	Informes, esquemas DB2	Se vincula con IEW y Excelerator; predicciones de rendimiento
ER-Designer	Chen and Associates (504-928-5765)	PC	ER/Rel	Diagramas ER + detalles, o form. texto	Si	No	DB2, ORACLE Ingres, IDMS/R DB2	Funciones de retroingeniería
ERwin	Logic Words (609-683-0054)	PC (Windows)	ER/Rel	Diagramas IDEF + detalles textuales	Si	No	DB2	Jerarquías ISA, presentación personalizable, migración de claves ajenas
IDEF/Leverage	D. Appleton Co. (800-322-6614)	PC+MF (IBM, DEC)	ER/Rel	Diagramas IDEF	Si	Si	DB2, Supra, RDB, ORACLE, Ingres	Diccionario de datos compartido, integración de vistas, enfoque de tres esquemas
Lyddia	Cascade Software (617-862-6746)	PC	Asociativo/ Rel, Archivo plano	Diagramas vistos como «capítulos» de libro	Si	No	ORACLE, Ingres, DB2, dBase, Focus, Informix, documento de req. de datos	Metáfora de libro y capítulos; genera automáticamente diagramas ER a partir del libro

Tabla 15.1.Bis Sistemas orientados al diseño de bases de datos (*continuación*)

Sistema de diseño	Compañía (teléfono)	Trabaja en	Modelos conc./lóg.	Entrada del sistema	Normalización	Modelados de transacciones	Salidas del sistema	Comentarios sobre características del sistema
Master Plus	GESI S.r.l. (Italia: 06-3725278)	PC	ER/ Rel	Diagramas ER y de flujo de datos + detalles, usuarios, sucesos	Sí	Sí	dBase, ADABASE, ORACLE, DB2, S/38, AS/400, Informix, Rbase	Integración de vistas, metadatos vistos como esquema ER, generación de prototipos
PCIAST	Control Data Corporation (N/D)	PC	NIAM/ (N/A)	(N/A)	Sí	(N/A)	Disenios lógicos	(N/A)
Re-Engineering Product Set	Bachman Information Systems (617-273-9563)	PS/2 + MF	Sociedad/ Rel, Redes	VSAM, arch. sec., IDMS, IMS, DB2	Sí	Sí	DDL físico para DB2	Consejero experto, retroingeniería
RIDL*	Intellibase (Bélgica 32-3-324-5956)	HP-Apollo, VAX/VMS, Sun Sparc	NIAM/ Rel	Diagramas NIAM + detalles, DFD	Sí	No	Ingres, ORACLE, DB2, Sybase, ISO SQL2	Desnormalización, mapas de transformación para esquemas, base de reglas, sólidas capacidades para modelar restricciones
Silverrun	XA Systems (800-848-3359)	Mac, PC	ER/ Rel	Diagramas ER de Merise + detalles	Sí	Un poco	DDL de SQL genérico, generación de pantallas e informes	Mecanismo para definir vistas, diccionario de datos compartido

N/A = no disponible.

Tabla 15.2. Sistemas CASE más generales que pueden diseñar bases de datos

Sistema de diseño	Compañía (teléfono)	Trabaja en	Modelos conc./lóg.	Entrada del sistema	Normalización	Modelados de transacciones	Salidas del sistema	Comentarios sobre características del sistema
Automate Plus (y Systems Engineer)	LBMS, Houston, TX (800-231-7515)	PC	ER/ Rel	Diagramas ER, DFD, otros diagramas	Sí	Sí	DB2, IDMS, ADABAS, Ingres, Informix, ORACLE	Apoyo a SSADM y SADT, diccionario de datos compartido, interfaz con AD/ Cycle Rep., prototipos y gestión de proyectos (producto Sys.Eng.)
Familia de productos CASE	ORACLE (800-672-2531)	Unix, PC, VAX/VMS	ER/ Rel	Diagramas ER y otros	Sí	Sí	ORACLE, DB2	Generador de aplicaciones, desarrollo estructurado, jerarquías de generalización
DEFT	SQL Systems (617-270-4150)	DEC VAX VMS/Unix; Mac	ER/ Rel	Diagramas ER, DFD, diagramas de estructura + definiciones de form.	No	No	Sybase, ORACLE, RDB, Ingres, Informix	Funciones de retroingeniería; compuerta VAX, gestión de vistas
Design/IDEF	Meta Software Cambridge, MA (800-227-4106)	Mac PC (Windows), Unix	ER/ ninguno	Diagramas IDEF, SADT	No	Sí	Informes	Diccionario de datos interactivo
DesignAID II	Transform Logic Scottsdale, AZ (800-872-8296)	PC	ER/ (Rel)	Diagramas ER + detalles	Sí	No	Informes	Hincapié en análisis estructurado
EasyCASE Plus	Evergreen CASE Tools (206-881)-5149)	PC	ER/ (Rel)	Diagramas ER, DFD, otros	No	No	Informes	Diccionario de datos compartido, se conserva en formato dBase
ENTERPRISE	Computer Associates (617-329-7700)	(N/A)	(N/A) Rel, Ntwk	(N/A)	Sí	Sí	Esquemas, formularios, informes	Múltiples herramientas, diseños físico, depósito compartido

N/A = no disponible.

Tabla 15.2.Bis Sistemas CASE más generales que pueden diseñar bases de datos (*continuación*)

Sistema de diseño	Compañía (teléfono)	Trabaja en	Modelos conc./lóg.	Entrada del sistema	Normalización	Modelados de transacciones	Salidas del sistema	Comentarios sobre características del sistema
Excelator	Index Technology (617-494-8200)	PC, Sun, Apollo, VAX-station	ER/ Rel., Redes	Diagramas ER + detalles, restricciones, vistas, otros diagramas	Si	Si	DB2	Personalización, diseño físico, interfaces con AD/Cycle Repository y VAX CDD/Plus
Foundation (DESIGN/I)	Arthur Andersen (312-507-5151)	PC + IBM MVS, TS0, CICS	ER/ Rel	Diagramas ER + detalles, otros diagramas	Si	Si	DB2	Diccionario de datos como alternativa de AD/Cycle, funciones de prototipos
Information Engineering Facility (IEF)	Texas Instruments (214-575-4404)	PC + IBM MVS Unix	ER/ Rel	Diagramas ER de Martin, diagr. de jerarquía de entidades, otros	Si	Si	DB2, Oracle, ORACLE, RDB, SQL/DS, Sybase, Ingres	Jerarquías de generalización, ciclo de vida completo, enciclopedia de macrocomputador
Information Engineering Workbench (IEW)	Knowledge Ware (800-338-4130)	PC+, IBM MVS, TS0, CICS	ER/ Rel, Jer	Diagr. ER + detalles, otros diagramas	A 1NF	Si	ORACLE, DB2, IMS, informes	Basado en sistema experto, neutral resp. a metodología, enciclopedia compartida, funciones de retroingeniería
POSE	Computer Systems Advisors (201-391-6500)	PC	ER/ Rel	Diagr. ER + detalles, transacciones	Si	Si	DB2, SQL/DS Focus, AS/ 400 ADABAS	Dicc. de datos compartido, conjunto de herramientas modular, diccionario compartido
PREDICT CASE	Software AG (Germany 06151-5040)	Macrocomp + est. trab.	ER/ Rel + gpos inf	Diagr. ER + detalles, modelos de funciones	No	Como «funciones del sistema»	ADABAS, informes, diagramas	Dicc. de datos compartido, orientado a obj., I/F a 4GL natural, jerarquías de generalización
PSL/PSA	Meta/LBMS (800-333-META)	VAX/VMS Pyramid/ Unix PC/ Unix V, MF	(ER)/ Rel	Especificaciones	Si	No	DB2, SQL genérico, informes	Metamodelo de objetos, propiedades, papeles, interrelaciones; integración de vistas

(Continúa en la siguiente página)

Tabla 15.2.2Bis Sistemas CASE más generales que pueden diseñar bases de datos (*continuación*)

Sistema de diseño	Compañía (teléfono)	Trabaja en	Modelos conc./lóg.	Entrada del sistema	Normalización	Modelados de transacciones	Salidas del sistema	Comentarios sobre características del sistema
Software through Pictures	Interactive Development Environments (415-543-0900)	Unix (Sun, DEC, HP)	ER/Rel, Jef	Diagramas ER + detalles, DFD, otros diagramas	No	No	DB2, Informix Ingres, Interbase, Sybase, Troll/ USE, ORACLE, ANSI SQL	Editores gráficos, visualización, diccionario compartido orientado a plantillas
Sylva Foundry	CADWARE (203-387-1853)	PC	ER/Rel, orientado a objetos	Diagramas ER, DFD	No	No	Informes	Editor basado en reglas, dicc. datos compartido, el modelo lógico es Schlar-Mellor (Rel+ER), diálogos, paneles, personalización con Sylva/Foundry
System Architect	Popkin Software, N.Y. (212-571-3434)	PC (OS/2, Windows)	ER/Rel	Diagramas ER + detalles	Si	Si	DB2, ORACLE, SQL genérico, IDMS, informes	Reglas en línea, interfaz de hoja de cálculo, diccionario de datos extensible en formato dBase, jerarquías de generalización, «sincronización» de objetos arbitrarios
Teamwork	Cadre (+ Ingres) (401-351-5950)	Estaciones de trabajo (DEC, HP, Apollo, IBM, Sun), DEC, VAX, VMS/Unix	ER/Rel	Diagramas ER, DFD, otros diagr.	Si	Si	Ingres y otros	Desnormalización, dominios, diccionario de datos abierto, análisis estructurado

ticas sobresalientes de cada sistema, pero, dada la amplitud de la cobertura y la rapidez del cambio en el ámbito comercial, podría haber algunas inexactitudes. Si al lector le interesa algún sistema, puede consultar al proveedor para obtener mayor información; las descripciones que siguen son apenas un bosquejo. Los números de teléfono se incluyen sólo para conveniencia del lector; puede que no sean vigentes.

Siguiendo la misma división de los apartados 15.7 y 15.8, se han dividido estos sistemas en aquellos orientados específicamente al diseño de bases de datos y en sistemas CASE más amplios que incluyen herramientas de diseño de bases de datos.

15.10. Limitaciones de los entornos de diseño automatizado de bases de datos

El apartado 15.3 trató la falta de concordancia general entre las metodologías y las herramientas automatizadas. En este apartado se hará un análisis más específico de los problemas y limitaciones de las herramientas disponibles en la actualidad. Por supuesto, ninguna herramienta individual posee todos estos defectos.

Un problema importante que surge con frecuencia en el caso particular de las herramientas CASE es que el apoyo de funciones/procesos está más desarrollado que el apoyo del diseño centrado en los datos. Cuando están presentes el modelado de datos y el de funciones, es difícil relacionarlos entre sí, por lo que el análisis conjunto de datos y funcional (descrito en el capítulo 9) se convierte en un reto logístico.

Muchos sistemas ofrecen un apoyo bastante desigual a lo largo de las diferentes fases del proceso de diseño. Dependen de un conjunto de diagramas bastante independientes, sin una metodología subyacente coherente. Son deficientes en cuanto a la descripción y presentación de posibilidades al diseñador; algunos sistemas no ofrecen ninguna medida o estimación del rendimiento. Con raras excepciones, hay poca visualización del progreso a lo largo del proceso de diseño, o seguimiento, por parte del sistema, de diseños alternativos.

A menudo faltan herramientas importantes y fases del diseño. Por ejemplo, ciertos sistemas carecen de apoyo para la normalización, integración de vistas, modelado de transacciones y combinaciones de transacciones, o diseño físico de bases de datos. Algunos son incapaces de gestionar restricciones, como las cardinalidades de las interrelaciones o las declaraciones de campos no nulos.

Si bien casi todos los sistemas permiten al diseñador trazar diagramas y almacenarlos en un diccionario del sistema, algunos no realizan el proceso inverso de generar los diagramas a partir de los metadatos almacenados. Asimismo, muchos sistemas tienen límites gráficos de la diagramación. Es común encontrar límites en el número de objetos y el número de conectores, límites en

el número de niveles en los que se pueden detallar los objetos, carencia de recursos para mover grupos de objetos y definiciones bajas de pantalla e impresora para la visualización de diagramas. Además, los gráficos se usan de forma poco imaginativa en la mayoría de los sistemas. El objetivo final es ayudar al diseñador en la exploración, descubrimiento y refinamiento de la estructura de base de datos. La mera reproducción en la pantalla de diagramas trazados en papel es un uso muy limitado de los recursos de los computadores.

Además, los sistemas actuales presentan varios problemas sutiles, que se hacen más evidentes —y problemáticos— una vez que los diseñadores han utilizado el sistema durante algún tiempo. En primer lugar, la capacitación no es trivial; presenta una larga curva de aprendizaje, sobre todo en los sistemas más amplios. En segundo lugar, los diseñadores han encontrado que, si bien las herramientas automáticas colaboran en cometidos simples, producen menores beneficios en el caso de tareas más complejas. Quizá esto vaya unido a las dificultades de compartir datos en los diseños grandes. En tercer lugar, a menudo se desarrolla una tensión entre la creatividad y el rigor al diseñar bases de datos. Es difícil desarrollar software siguiendo las reglas de otros, y esto es, efectivamente, lo que un diseñador debe hacer en la mayoría de los sistemas de diseño de bases de datos. Finalmente, los diseños de sistemas son ayudas de producción, pero existe con frecuencia un equilibrio inquietante entre la productividad y la calidad. Quizá el enfoque japonés tenga mérito en el diseño automatizado de bases de datos: esforzarse por lograr la calidad, y la productividad vendrá a continuación.

15.11. Tendencias en los entornos de diseño automatizado de bases de datos

Las herramientas y los entornos automatizados para el diseño de bases de datos continuarán evolucionando, impulsados por los avances en el hardware, por los paradigmas innovadores de software, por mejoras en la tecnología de interfaces para el usuario y, desde luego, por nuevas direcciones y refinamientos en las metodologías del diseño. Es de esperar que las limitaciones descritas en el apartado 15.10 desaparezcan con el tiempo. En seguida se resumen algunas tendencias y direcciones, muchas de las cuales comienzan a aparecer en los sistemas expuestos en los apartados 15.6 al 15.8. Esta no es una lista exhaustiva.

15.11.1. Sistemas expertos y basados en reglas

Los sistemas expertos integrados se utilizarán cada vez con más frecuencia para personalizar los sistemas de diseño y guiar a los diseñadores. Si bien los conocimientos metodológicos y los consejos prácticos de diseño pueden estar codificados en procedimientos o de alguna otra manera incorporados en un sistema,

tiene muchas ventajas almacenar esta información en forma de reglas: los conocimientos se hacen explícitos, visibles, rigurosos, modificables y extensibles. Se puede usar mecanismos generales de inferencia que sean independientes de cualquier cambio de la base de reglas. Las reglas pueden ayudar a detectar acciones incongruentes de la base de datos, mejorar el apoyo analítico mediante una comparación flexible de patrones, y mantener diálogos adaptables con los usuarios. Las mejores aplicaciones para una tecnología basada en reglas son aquellas en las que los conocimientos son de bajo nivel y un tanto especializados, pero precisos. Esto incluye las siguientes áreas:

1. Guía y supervisión metodológica (personalización, adaptación de rutas de acceso y conjuntos de herramientas, guía y sugerencias para los pasos que siguen). Por ejemplo, una herramienta podría presentar un conjunto de primitivas descendentes en forma de menú, como hace el sistema COMIC (Kangassalo, 1989).
2. Análisis de los esquemas en varios niveles (apoyo para las transacciones, compleción de los esquemas conjuntos de datos y funciones, actualización de esquemas, seguridad de esquemas).
3. Diseño lógico (normalización, desnormalización, selección de claves primarias, división y fusión de entidades, decisiones sobre el almacenamiento de datos derivados).
4. Disposición de diagramas (minimización de las longitudes y superposición de los conectores, agrupamiento para mejorar la legibilidad, mantenimiento de las jerarquías de entidades, colocación de las etiquetas de texto).
5. Definición y uso de perfiles personalizados de usuario (niveles de habilidad [novato, intermedio, experto], preferencias de presentación y edición, directorios de trabajo, estilo de informes de herramienta, frecuencia de grabación).
6. Diseño físico (diseño de campos, decisiones de agrupamiento, métodos de acceso y tamaño). Como se puntuó antes en este libro, el diseño físico es muy específico respecto al sistema, así que las reglas para el diseño físico deben adaptarse a entornos de DBMS específicos.

15.11.2. Retroingeniería

La mayoría de los sistemas de diseño actuales están orientados al diseño a partir de cero. Sin embargo, gran parte del diseño de base de datos es en realidad *re-diseño*. Las herramientas para la retroingeniería se harán en el futuro cada vez más comunes (Chikovsky y Cross, 1990). Estas herramientas comienzan captando definiciones de bases de datos existentes de DBMS y archivos planos, y a veces también captan el código fuente de la aplicación acompañante. Una vez introducida en el sistema la información existente, puede analizarse y presentarse, muchas veces en un nivel más alto de abstracción. Por ejemplo, un esquema IMS existente se puede leer, analizar y transformar en el diagrama ER co-

rrespondiente (Navathe y Awong [1987] presentan un posible procedimiento; véase también el apartado 14.5). En este punto, el usuario puede incluir nuevos requisitos, extender el diseño, integrarlo con otros diseños y migrar hacia un DBMS nuevo, o incluso a una plataforma nueva. De hecho, el diseñador es libre de combinar los dos enfoques de ingeniería, hacia atrás y hacia adelante, reflejando —en un nivel metodológico más amplio— los enfoques mixto y centrífugo que vimos para el diseño del esquema conceptual.

Desafortunadamente, el uso de retroingeniería para producir un *esquema* nuevo sólo resuelve parte del problema. Idealmente, las herramientas serán capaces de generar un plan de reorganización o serie de mandatos que trasladarán automáticamente al nuevo esquema los *datos* existentes, posiblemente en un entorno nuevo. Además, tendrán que considerar la modificación o regeneración de los programas de aplicación existentes para utilizar la base de datos reestructurada.

15.11.3. Herramientas integradas y mejoradas para el diseño de bases de datos

Los conjuntos de herramientas de diseño de bases de datos llegarán a estar integrados tanto funcional como arquitectónicamente (como se describió en el apartado 15.2.3). Un avance muy necesario es que también estén más integrados con las herramientas de análisis estructurado. La mayor parte de los sistemas de diseño actuales tienen un enfoque simplista al respecto (por ejemplo, hacer corresponder todo un almacén de datos con un tipo de entidades). En el futuro, empero, la correspondencia entre datos y perspectivas funcionales se producirá con mucho más detalle a través del diccionario de datos y procesos.

Establecimos en el apartado 15.2.3 que las herramientas necesitan ser robustas, es decir, capaces de hacer frente a esquemas tanto malos como buenos, o a esquemas con información ausente. Las herramientas serán también más *prácticas*; afrontarán mejor la complejidad del diseño de bases de datos del mundo real. No es suficiente que una herramienta dibuje imágenes de entidades, interrelaciones y atributos, que genere el equivalente relacional básico y que considere el trabajo ya realizado. Las herramientas tendrán que especificar restricciones, políticas organizativas intrincadas y cargas de transacciones. Generarán esquemas para servidores de bases de datos con disparadores, procedimientos almacenados, máquinas de inferencia integradas y capacidades de bases de datos distribuidas y orientadas a objetos (véanse los apartados 15.11.6 y 15.11.7).

Las herramientas dejarán de ser los cuadernos de notas gráficas, pasivos y relativamente débiles en semántica que son hoy para convertirse en *socios activos del diseño*. Las herramientas generarán y presentarán alternativas de diseño a sus usuarios en el momento adecuado y en un contexto pertinente. Las alternativas irán acompañadas de análisis de posibilidades e impacto así como

implicaciones de rendimiento. Al principio del proceso de diseño, las herramientas usarán la heurística para «echar a andar» el diseño, infiriendo el esquema inicial a partir de formularios (Choobiner et al., 1988) o infiriendo dependencias funcionales significativas a partir de casos de datos existentes (Bitton, Millman y Torgersen, 1989). Por último, las herramientas se harán más fuertes en áreas concretas, como el modelado de transacciones y la integración de vistas.

15.11.4. Interfaces mejoradas para el usuario

Los computadores personales y las estaciones de trabajo se orientan cada vez más hacia las interfaces gráficas de usuario (GUI). Aunque Macintosh fue un innovador en este campo, los PC de IBM y las máquinas compatibles tienen ahora el OS/2 Presentation Manager (y el sistema de transición Windows encima de DOS) y las estaciones de trabajo tienen también rostros gráficos de multiventanas. Estas interfaces presentan galerías de iconos para elegir acciones, objetos o detalles de presentación. El **empleo directo** es común: el usuario puede arrastrar y soltar iconos sobre otros objetos con el ratón para iniciar las acciones. El paradigma Hypercard en Macintosh permite construir **pilas** personalizadas, que pueden incluir sonido e imágenes arbitrarias de mapa de bits. Están apareciendo paquetes de herramientas completas de desarrollo de UI, como NeXTStep en la máquina NeXT, que apoyan la rápida creación de prototipos de interfaces de usuario. Dada la dificultad de escribir para sistemas gráficos, existen ya arquitecturas y paquetes de herramientas (como Motif de la Open Software Foundation) que ayudan a los creadores de herramientas a transportar las interfaces para usuario final de un sistema a otro.

La mayoría de los sistemas de diseño de bases de datos son ahora gráficos y utilizarán las GUI cada vez más en el futuro (véase *Proceedings, IEEE Workshop on Visual Languages* [1988] donde se trata un área afín). Los gráficos son una ayuda para la comprensión a través de la visualización y el descubrimiento; la mitad de la batalla de crear un sistema de bases de datos es ayudar al diseñador a que comprenda y asimile los diseños en evolución. Estarán presentes múltiples niveles de ventanas para ayudar a la revelación progresiva de los detalles cuando se necesiten. Para facilitar mejor el cambio a un contexto afín, los sistemas pueden contener redes de enlaces de hipertexto (Conklin, 1897). Se refinarán mecanismos para presentar grandes diagramas y permitir la navegación a través de ellos. En vez de limitar las presentaciones en pantalla a diagramas ER estándar y más bien mediocres, los sistemas usarán colores, sombreado, realce, parpadeo, tamaños variables de objetos y otros dispositivos gráficos para destacar objetos o regiones, centrar la atención del diseñador, comunicar semántica adicional y delinejar opciones y concesiones durante todo el proceso de diseño.

15.11.5. Posibilidad de personalización

Los sistemas actuales de diseño permiten una personalización bastante simple de características tales como estilos de diagramas y perfiles de usuarios. Los sistemas futuros serán bastante más personalizables, adaptándose a una amplia gama de enfoques. Una forma de conseguir esto será por medio de **elecciones y opciones** ofrecidas en diversos puntos del proceso de diseño. Por ejemplo, un sistema que se está diseñando ahora en DEC propone usar ER extendido, NIAM e IDEF1-X como modelos conceptuales alternativos. Muchos sistemas actuales apoyan la generación de esquemas para varios DBMS objetivo, y esta flexibilidad aumentará ciertamente con el tiempo.

Una segunda forma importante de hacer más personalizables los sistemas es hacerlos **orientados a tablas** y permitir que las tablas sean modificadas. El componente Customizer de Excelerator es un ejemplo de esta dirección; permite que se añadan, sin que se noten las costuras, nuevas entidades, interrelaciones, tipos de gráficas y menús. Por último, la capacidad de personalización puede mejorarse por medio de mecanismos de extensión de propósito general, como **reglas y políticas**. Por ejemplo, una organización podría formular sus propias reglas para determinar cuándo debe ser considerada la desnormalización al hacer el diseño físico de las bases de datos. El concepto de políticas de AD/Cycle permite la derivación personalizada de atributos de objetos, y disparadores generales para propagar cambios entre objetos en el depósito.

15.11.6. Herramientas para el diseño de bases de datos distribuidas

Las **bases de datos distribuidas** son un fenómeno relativamente reciente. En ellas, las tablas de bases de datos son fraccionadas y los **fragmentos** se almacenan en múltiples lugares, a menudo con algunos datos almacenados redundantemente en varias posiciones⁴. Es una ventaja poner los datos donde se utilizan y tener disponibles, a nivel local, datos que son fiables (lo que se conoce como **autonomía local**). Las bases de datos distribuidas se adaptan bien a estructuras organizativas descentralizadas, proporcionan un buen rendimiento y tienen arquitecturas escalables. Los usuarios finales no necesitan estar enterados de la posición real de los datos (**transparencia de la ubicación**) o de las complejidades de optimizar el acceso a los datos distribuidos. Por otra parte, puede ser una pesadilla asegurar la integridad de los datos distribuidos y administrar un sistema de bases de datos distribuidas.

Cuando las bases de datos distribuidas se hagan más comunes, habrá un apoyo automatizado creciente para su diseño (Teorey, 1989; Navathe y Ceri,

⁴ Los servidores de base de datos a los que se tiene acceso a través de la arquitectura cliente/servidor pueden contar o no con apoyo de bases de datos distribuidas. El simple hecho de que se llegue a los servidores a través de la red no hace que las bases de datos en sí estén distribuidas; de hecho, casi siempre están almacenadas en el servidor. Incluso si los servidores pueden actuar como **compuertas** de otros DBMS, la base de datos en sí no se considerará necesariamente distribuida.

1985; Navathe et al., 1990). Las herramientas ayudarán al usuario a fragmentar las relaciones tanto horizontal como verticalmente y a colocar los fragmentos en lugares distintos. Un ejemplo de fragmentación horizontal sería colocar los registros de los empleados de una compañía en Los Angeles, Chicago y Boston en esas tres localidades, respectivamente. La fragmentación vertical (que se parece mucho a la normalización y descomposición) podría implicar guardar la parte administrativa del historial médico de un paciente (por ejemplo, número de teléfono y dirección) en una posición y distribuir la parte médica (por ejemplo, grupo sanguíneo, fecha de nacimiento) a otra parte. Las herramientas ayudarán también a determinar qué fragmentos necesitan ser reproducidos para acelerar el acceso y para visualizar topologías distribuidas.

Se presentan más desafíos cuando la base de datos distribuida abarca DBMS heterogéneos, quizá en diferentes modelos de datos. Las herramientas de integración de vistas serán útiles para producir una vista conceptual global de las bases de datos individuales y resolver incongruencias entre ellas.

15.11.7. Sistemas orientados a objetos

Los sistemas de diseño de bases de datos serán afectados por la relativamente reciente invasión de los sistemas orientados a objetos en tres importantes áreas. El cambio más radical puede venir del área de **entornos de desarrollo orientados a objetos** (por ejemplo, Smalltalk [Goldberg y Robson 1983], Eiffel [Meyer 1988], C++, Objective C, y OS/2 Presentation Manager), que pueden utilizarse para hacer diseño de aplicaciones. En estos entornos, la programación se realiza definiendo **objetos** y comunicaciones entre objetos. Los objetos pueden ser cualquier cosa: tareas, procesos, elementos de datos, mensajes, clases y objetos compuestos. La comunicación no es secuencial; los objetos envían mensajes y realizan tareas cuando es necesario. Conceptos clave son el **encapsulamiento** (se oculta la naturaleza interna de los objetos) la **herencia** (las clases de objetos están en una jerarquía y pueden heredar conducta de sus padres), la **especialización** (una nueva clase de objetos sólo necesita ser definida en términos de cómo difiere de su clase padre) y la **persistencia** (los objetos no son necesariamente temporales, sino que pueden permanecer hasta ser destruidos explícitamente). Puesto que los objetos incorporan aspectos tanto de las estructuras de datos persistentes como de los programas (**métodos de objetos**) que usan estas estructuras, es práctico diseñar simultáneamente los objetos y sus métodos. En efecto, esto desplaza al diseñador de bases de datos hacia la esfera de la programación.

La segunda área relacionada es la de los **sistemas de gestión de bases de datos orientados a objetos** (OODBMS, *Object Oriented Database Management Systems*) (por ejemplo, Ontos, Object/1, O₂, Exodus [Carey et al., 1986]). Los OODBMS combinan algunas de las funciones de los DBMS tradicionales (por ejemplo, gestión de almacenamiento secundario, transacciones y optimización de las consultas) con las capacidades de los sistemas orientados a objetos (por ejemplo, encapsulamiento, objetos complejos, jerarquías de tipos, herencia, ex-

tensibilidad, etc.) (véase Atkinson et al., 1989; Dawson, 1989). Para los OODBMS, el diseño de bases de datos se convierte en diseño de objetos, acoplado con el diseño de métodos dentro de los objetos. Las herramientas de diseño para este entorno diferirán considerablemente de las que apoyan el diseño de bases de datos relacionales.

El área final de impacto es que los sistemas de diseño de bases de datos mismos serán *construidos* utilizando paradigmas orientados a objetos. Esto ya está empezando a suceder con las interfaces para usuario en entornos gráficos de multiventanas. Un importante beneficio desde el punto de vista del usuario es que un sistema así es muy *extensible*. Los objetos definidos por el usuario como nuevos tipos de diagramas y reglas metódicas pueden ser añadidos al sistema sin que se noten las costuras y con un mínimo de esfuerzo.

15.11.8. Diseño automático de esquemas a partir de casos

La herramienta definitiva de diseño de esquemas de bases de datos sería capaz de construir el esquema de la base de datos sabiendo qué casos tendría que modelar. Más aún, podría ajustar o adaptar el esquema según aparecieran nuevos casos. En aplicaciones de bases de datos más recientes, como el diseño asistido por computador (CAD) o la fabricación asistida por computador (CAM) hay muchos tipos, pero posiblemente sólo unos cuantos casos por tipo. En tales situaciones sería ventajoso que el sistema pudiera construir automáticamente el esquema de una base de datos. Un problema relacionado surge cuando un gran volumen de texto es representado en una base de datos. En este caso una herramienta podría construir automáticamente una red de conceptos detectando conceptos del texto. La investigación de tales problemas se desarrolla actualmente, basándose en aplicar ideas de inteligencia artificial (Beck et al., 1990; Anwar et al., 1990).

Ejercicios

Los ejercicios de este capítulo se relacionan con los presentados en capítulos anteriores; la diferencia reside en el uso de herramientas automatizadas. Utilizando un sistema de diseño automatizado al que tenga acceso, siga los métodos dados en capítulos anteriores de este libro para realizar los siguientes ejercicios.

- 15.1. Diseñe algunas de las vistas de los ejercicios del capítulo 4.
- 15.2. Diseñe algunos de los esquemas funcionales de los ejercicios del capítulo 8.
- 15.3. Realice un análisis conjunto de datos y funciones, como se pedía en los ejercicios del capítulo 9.
- 15.4. Cree un diseño lógico de alto nivel, como se pedía en los ejercicios del capítulo 11.

-
- 15.5. Haga corresponder un diseño ER con uno relacional, como se pedía en los ejercicios del capítulo 12.
 - 15.6. Haga corresponder un diseño ER con diseño de redes o jerárquico, como se pedía en los ejercicios del los capítulos 13 y 14.
El ejercicio final es amplio pero importante.
 - 15.7. Examine un sistema de diseño de su organización o uno descrito en trabajos publicados y analice cómo se apoya la metodología explicada en este libro. Sugiera formas en que el sistema y sus herramientas componentes podrían ser mejoradas, ampliadas o modificadas para apoyar la metodología de este libro.

Bibliografía

- A. Albano, et al., ed., *ComputerAided Database Design: The DATAID Project*, North-Holland, 1985.
- T.M. Anwar, S.B. Navathe y H. W. Beck, «SAMI: A Semantically Adaptive Modeling Interface for Schema Generation over Multiple Databases», informe técnico, Database Systems R&D Center, University of Florida, Gainesville, Fla., 1990.
- M. Atkinson, F. Bancilhon, D. Dewitt, K. Dittrich, D. Maier y S. Zdonlc. «The Object-Oriented Database System Manifesto». En *Proc. Conference on Deductive and Object Oriented Databases*, Kyoto, 1989.
- C. Bachman, «A CASE for Reverse Engineering», *Datamation*, 1.º de julio de 1988, 49-56.
- C. Batini y M. Lenzerini, «A Methodology for Data Schema Integration in the Entity-Relationship Model», *IEEE Transactions on Software Engineering*, SE10, núm. 6, noviembre de 1984, 650-63.
- C. Batini, M. Lenzerini y S. B. Navathe, «Comparison of Methodologies for Database Schema Integration», *ACM Computing Surveys*, 18, núm. 4, diciembre de 1986, 323-64.
- H. W. Beck, T. M. Anwar y S. B. Navathe, «Towards Database Schema Generation by Conceptual Clustering», informe técnico, Database Systems R&D Center, University of Florida, Gainesville, Fla., 1990.
- D. Bitton, J. Millman y S. Torgersen, «A Feasibility and Performance Study of Dependency Inference», *Proc. IEEE Data Engineering Conference*, Los Angeles, 1989.
- M. Bouzeghoub, G. Gardarin y E. Metais, «Database Design Tools: An Expert System Approach». En *Proc. Ninth International Conference on Very Large Databases*, Estocolmo, 1985.
- R. P. Braegger, A. M. Dudler, J. Rebsamen y C. A. Zehnder, «Gambit: An Interactive Database Design Tool for Data Structures, Integrity Constraints and Transactions», *IEEE Transactions on Software Engineering*, SE-11, núm. 7, julio de 1985, 574-83.
- M. Carey et al., «The Architecture of the Exodus Extensible DBMS», *Proc. First Annual Workshop on OODBMSs*, ACM, 1986.
- T. Catarci y F. M. Ferrara. «OPTIM_ER: An Automated Tool for Supporting the Logical Design within a Complete CASE Environment». En C. Batini, ed., *Proc. Seventh International Conference on Entity Relationship Approach*, Roma, North-Holland, 1988.

- S. Ceri, ed., *Methodology and Tools for Database Design*. Elsevier Science North-Holland, 1983.
- E. J. Chikovsky y J. Cross, «Reverse Engineering and Design Recovery: A Taxonomy», *IEEE Software*, enero de 1990, 13-18.
- J. Choobineh et al., «An Expert Database Design System based on Analysis of Forms», *IEEE Transactions on Software Engineering*, 4, núm. 2, febrero de 1988, 242-53.
- J. Conklin, «Hypertext: An Introduction and Survey», *IEEE Computer*, septiembre de 1987, 17-41.
- J. Dawson, «A Family of Models [OODBMSs]», *Byte*, septiembre de 1989, 277-86.
- O. de Troyer, «RIDL*: A Tool for the Computer Assisted Engineering of Large Databases in the Presence of Integrity Constraints», En *Proc. ACM-SIGMOD International Conference on Management of Data*, Portland, Ore., 1989.
- O. de Troyer, R. Meersman y R. Verlinden, «RIDL* on the CRIS Case: A Workbench for NIAM». En T. W. Olle, A. A. VerrijnStuart y L. Bhabuta, eds., *Computerized Assistance During the Information Systems Life Cycle*, Elsevier Science (North-Holland), 1988, 375-459.
- A. Dogac et al., «A Generalized Expert System for Database Design», *IEEE Transactions on Software Engineering*, 15, núm. 4, abril de 1989, 479-91.
- R. Elmasri, J. Larson y S. B. Navathe, «Schema Integration Algorithms for Federated Databases and Logical Database Design», Honeywell Computer Sciences Center Technical Report #CSC 86-9:8212, Golden Valley, Minn., 55427, 1986.
- R. Elmasri, J. Weeldryer y A. Hevner, «The Category Concept: An Extension to the Entity-Relationship Model», *International Journal on Data and Knowledge Engineering*, 1, núm. 1, mayo de 1985.
- A. S. Fisher, *CASE—Using Software Development Tools*, John Wiley and Sons, 1988.
- J. T. Fisher, «IBM's Repository», *DBMS*, 3, núm. 1, enero de 1990, 42-49.
- C. Gane, *Computer-Aided Software Engineering: the Methodologies, the Products, and the Future*, Prentice Hall, 1990.
- A. Goldberg y D. Robson, *Smalltalk-80: The Language and Its Implementation*, Addison Wesley, 1983.
- A. Hazah, «Data Dictionaries: Paths to a Standard», *Database Programming & Design*, 2, núm. 8, agosto de 1989, 26-35.
- A. Hazah, «Making Ends Meet: Repository Manager», *Software Magazine*, 9, núm. 12, diciembre de 1989, 59-71.
- Index Technology Corp, «Excelerator». En *Proc. CASE Symposium*, Digital Consulting, 1987.
- H. Kangassalo, «COMIC: A System for Conceptual Modeling and Information Construction», *First Nordic Conference on Advanced Systems Engineering—CASE89*, Estocolmo, Suecia, 1989.
- C. McClure, *CASE Is Software Automation*, Prentice-Hall, 1989.
- D. Maier, J. D. Ullman y M. Y. Vardi, «On the Foundations of the Universal Relation Model», *ACM Transactions on Database Systems*, 9, núm. 2, junio de 1984.
- B. Meyer, *Object-Oriented Software Construction*, Prentice-Hall, 1988.
- J. Mylopoulos et al., «Information System Design at the Conceptual Level—The TAXIS Project», *IEEE Database Engineering*, 7, núm. 4, diciembre de 1984.
- S. B. Navathe y A. M. Awong, «Abstracting Relational and Hierarchical Data with a Semantic Data Model». En S. March, ed., *Proc. Sixth International Conference on Entity-Relationship Approach*, Nueva York, North-Holland, 1987.
- S. B. Navathe y S. Ceri, «A Comprehensive Approach to Fragmentation and Allocation

- of Data». En J. A. Larson, S. Rahimi, eds., *IEEE Tutorial on Distributed Database Management*, IEEE, 1985.
- S. B. Navathe, R. Elmasri y J. A. Larson, «Integrating User Views in Database Design», *IEEE Computer*, 19, núm. 1, enero de 1986, 50-62.
- S. B. Navathe et al., «A Mixed Partitioning Methodology for Distributed Database Design», informe técnico núm. 9017, Database Systems R. D. Center, University of Florida, Gainesville, Fla., 1990.
- B. Pace, «Learn-As-You-Go CASE [POSE]», *System Builder*, abril de 1989.
- M. Prabandham et al., «A View of the IRDS Reference Model», *Database Programming Design*, 3, núm. 3, marzo de 1990, 40-53.
- Proc. IEEE Workshop on Visual Languages*, IEEE Computer Society Press, 1988.
- P. T. Quang, «Merise: A French Methodology for Information Systems Analysis and Design», *Journal of Systems Management*, marzo de 1986, 21-24.
- D. Reiner, G. Brown, M. Friedell, J. Lehman, R. McKee, P. Rheingans y A. Rosenthal, «A Database Designer's Workbench». En S. Spaccapietra, ed., *Proc. Fifth International Conference on Entity-Relationship Approach*, Dijon, North-Holland, 1986.
- A. Rosenthal y D. Reiner, «Theoretically Sound Transformations for Practical Database Design». En S. March, ed., *Proc. Sixth International Conference on Entity-Relationship Approach*, Nueva York, North-Holland, 1987.
- A. Rosenthal y D. Reiner, «Database Design Tools: Combining Theory, Guesswork and User Interaction». En J. Lochovsky, ed., *Proc. Eighth International Conference on Entity Relationship Approach*, Toronto, North-Holland, 1989.
- A. P. Sheth, J. A. Larson, A. Cornelio y S. B. Navathe, «A Tool for Integrating Conceptual Schemas and User Views». En *Proc. of IEEE Fourth International Conference on Data Engineering*, Los Angeles, IEEE, 1988.
- V. Storey y R. Goldstein, «A Methodology for Creating User Views in Database Design», *ACM Transactions on Database Systems*, 13, núm. 3, septiembre de 1988, 305-38.
- T. Teorey, «Distributed Database Design: A Practical Approach and Example». *ACM-SIGMOD Record*, 18, núm. 4, diciembre de 1989.
- T. J. Teorey, *Database Modeling and Design: The Entity-Relationship Approach*, Morgan Kaufmann, 1990.
- T. Teorey, D. Yang y J. Fry, «The Logical Record Access Approach to Database Design», *Computing Surveys*, 12, núm. 2, 1980, 169-211.
- J. L. Williams, «Excelerator: A CASE Study», *Database Programming & Design*, 1, núm. 4, abril de 1988, 50-56.

Vocabulario técnico bilingüe*

TÉRMINO ORIGINAL EN INGLÉS	TÉRMINO USADO EN ESTE LIBRO	VOCABLOS ALTERNATIVOS DE USO COMÚN
<i>Actigram</i>	Actograma	—
<i>Affinity groups</i>	Grupos de afinidad	—
<i>Aggregates</i>	Agregados	—
<i>Aggregation Abstraction</i>	Abstracción de agregación	—
<i>Alias</i>	Alias (véase también Sinónimo)	—
<i>Alternate key</i>	Clave alternativa	Llave alternativa
<i>Analyze mode</i>	Modo analítico	—
<i>Analyzer</i>	Analizador	—
<i>Annotation</i>	Anotación (en diseños)	—
<i>Ansi/sparc architecture</i>	Arquitectura ansi/sparc	—
<i>Application requeriments</i>	Requerimientos de aplicaciones	Requisitos de aplicaciones
<i>Association</i>	Asociación	—
<i>Associativity</i>	Asociatividad	—
<i>Attribute</i>	Atributo	—

* Ante la falta de un lenguaje estandarizado en castellano para las ciencias de la computación, se ha elaborado el presente vocabulario con la traducción que hemos dado en este libro a los principales términos de la versión original en inglés.

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Attribute mandatory</i>	Atributo obligatorio	—
<i>Attribute non-null</i>	Atributo no nulo	—
<i>Bachman diagram</i>	Diagrama de Bachman	—
<i>Binary mapping</i>	Correspondencia binaria	—
<i>Binary Aggregation</i>	Agregación binaria	—
<i>Block</i>	Bloque	—
<i>BNF grammar</i>	Gramática BNF	—
<i>Bottom-up strategy</i>	Estrategia ascendente	—
<i>Bottom-up primitives</i>	Primitivas ascendentes	—
<i>Boyce-Codd normal form</i>	Forma normal de Boyce-Codd	—
<i>C language</i>	Lenguaje C	—
<i>C++ language</i>	Lenguaje C++	—
<i>Candidate key</i>	Clave candidata	Llave candidata
<i>Cardinality</i>	Cardinalidad	—
<i>Cardinality of a relation</i>	Cardinalidad de una relación	Cardinalidad de una interrelación
<i>Cardinality minimal</i>	Cardinalidad mínima	—
<i>Cardinality average</i>	Cardinalidad promedio	Cardinalidad media
<i>Cardinality maximal</i>	Cardinalidad máxima	—
<i>(CASE)</i>	Ingeniería de software asistida por computador	—
<i>Class subset</i>	Clases subconjunto	—
<i>Class generic</i>	Clase genérica	Clase paramétrica
<i>Classification Abstraction</i>	Abstracción de clasificación	Abstracción de ordenación
<i>Client/server architecture</i>	Arquitectura cliente/servidor	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Client/server</i>	Cliente/servidor	—
<i>Clustering</i>	Agrupamiento	—
<i>Collapsing one relation</i>	Integración en una relación	—
<i>Completeness of a primitive</i>	Compleción de una primitiva	Compleitud, lo completo
<i>Completeness</i>	Compleción	Compleitud, lo completo
<i>Completeness of a schema</i>	Compleción de un esquema	Compleitud de un esquema
<i>Compression</i>	Comprensión	—
<i>Computer</i>	Computador	Ordenador, computadora
<i>Concepts identical</i>	Conceptos idénticos	—
<i>Concepts compatibles</i>	Conceptos compatibles	—
<i>Concepts neighboring</i>	Conceptos vecinos	—
<i>Concepts balancing of</i>	Equilibrio de conceptos	Equilibrado de conceptos
<i>Concepts incompatible</i>	Conceptos incompatibles	—
<i>Concept mismatch</i>	Discrepancia de conceptos	No correspondencia de conceptos
<i>Concepts similarity</i>	Semejanza de conceptos	—
<i>Conceptual data model</i>	Modelo conceptual de datos	—
<i>Conceptual design</i>	Diseño conceptual	—
<i>Conceptual model</i>	Modelo conceptual	—
<i>Conceptual schema</i>	Esquema conceptual	—
<i>Conceptual level</i>	Nivel conceptual	—
<i>Concurrent access</i>	Acceso concurrente	—
<i>Conditional execution</i>	Ejecución condicional	—

TÉRMINO ORIGINAL EN INGLÉS	TÉRMINO USADO EN ESTE LIBRO	VOCABLOS ALTERNATIVOS DE USO COMÚN
<i>Conflict structural</i>	Conflictos estructurales	—
<i>Conflict analysis</i>	Ánalisis de conflictos	—
<i>Conflict name</i>	Conflictos de nombres	—
<i>Constrains</i>	Restricciones	—
<i>Control structure</i>	Estructuras de control	—
<i>Correctness of a schema</i>	Correcciones de un esquema	—
<i>Correctness of semantic</i>	Correcciones semánticas	—
<i>Correctness of syntactic</i>	Correcciones sintácticas	—
<i>Correctness</i>	Corrección	Exactitud
<i>Coupling</i>	Acoplamiento	—
<i>Coverage properties</i>	Propiedades de cobertura	—
<i>Coverage partial</i>	Cobertura parcial	—
<i>Coverage exclusive</i>	Cobertura exclusiva	—
<i>Coverage total</i>	Cobertura total	—
<i>Coverage overlapping</i>	Cobertura superpuesta	Cobertura solapada, recubierta
<i>Currency indicator</i>	Indicador de actualidad	—
<i>Currency</i>	Actualidad	—
<i>Cycles of relationships</i>	Ciclos de interrelaciones	—
<i>Check-in/check-out</i>	Comprobación de entrada/salida	—
<i>Child record type</i>	Tipo de registros hijo	—
<i>D-schema</i>	Esquema D	—
<i>Dangling entities</i>	Entidades colgantes	—
<i>Dangling subentities</i>	Subentidades colgantes	—
<i>Data item</i>	Elemento de datos	Elemento datos

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Data model hierarchical</i>	Modelo jerárquico de datos	Modelo de datos jerárquico
<i>Data model relational</i>	Modelo relacional de datos	Modelo de datos relacional
<i>Data dictionary</i>	Diccionario de datos	—
<i>Data schema</i>	Esquema de datos	—
<i>Data model</i>	Modelo de datos	—
<i>Data access</i>	Acceso a los datos	—
<i>Data and process dictionary</i>	Diccionario de datos y proceso	—
<i>Data store</i>	Almacén de datos	—
<i>Data volume</i>	Volumen de datos	—
<i>Data schemas</i>	Esquemas de datos	—
<i>Data dictionary systems</i>	Sistemas de diccionarios de datos	—
<i>Data system</i>	Sistema de datos	—
<i>Data active</i>	Dato activo	—
<i>Data structure</i>	Estructura de datos	—
<i>Data type</i>	Tipo de datos	—
<i>Data integrated</i>	Datos integrados	—
<i>Data local</i>	Datos locales	—
<i>Data model semantic</i>	Modelo semántico de datos	—
<i>Data migration</i>	Migración de datos	—
<i>Data definition language</i>	Lenguaje de definición de datos	—
<i>Data design</i>	Diseño de datos	—
<i>Database administrator</i>	Administrador de base de datos	Gestor de base de datos

TÉRMINO ORIGINAL EN INGLÉS	TÉRMINO USADO EN ESTE LIBRO	VOCABLOS ALTERNATIVOS DE USO COMÚN
<i>Database design</i>	Diseño de bases de datos	—
<i>Database</i>	Base de datos	—
<i>Database operation</i>	Operación de base de datos	—
<i>Database load</i>	Carga de bases de datos	—
<i>Databases integration</i>	Integración de bases de datos	—
<i>Dataflow</i>	Flujo de datos	—
<i>Dataflow diagram</i>	Diagrama de flujo de datos	—
<i>Dataflow model of</i>	Modelo de flujo de datos	—
<i>Datos database design conceptual</i>	Diseño conceptual de bases de datos	—
<i>DBMS</i>	Sistema de gestión de bases	—
<i>DC manager</i>	Administrador DC	Gestor DC
<i>Deadlock</i>	Bloqueo mutuo	—
<i>Decomposition diagrams</i>	Diagramas de descomposición	—
<i>Degree of a relation</i>	Grado de una relación	—
<i>Deletion anomaly</i>	Anomalía de eliminación	—
<i>Denormalization</i>	Desnormalización	—
<i>Dependency inclusion</i>	Dependencia de inclusión	—
<i>Dependency multivalued</i>	Dependencia polivalente	—
<i>Dependency functional</i>	Dependencia funcional	—
<i>Derived data</i>	Datos derivados	—
<i>Derived attribute</i>	Atributo derivado	—
<i>Design application programs</i>	Diseño de programas de aplicación	—
<i>Design data-driven</i>	Diseño orientado a los datos	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Design Function-driven</i>	Diseño Orientado a las funciones	—
<i>Design methodology</i>	Metodologías de diseño	—
<i>Design strategie</i>	Estrategia de diseño	—
<i>Design tree</i>	Arbol de diseño	—
<i>Design tools</i>	Herramientas de diseño	—
<i>Design tools heuristic</i>	Herramientas heurísticas de diseño	—
<i>Determinant</i>	Determinante	—
<i>Direct search</i>	Búsqueda directa	—
<i>Direct access</i>	Acceso directo	—
<i>Distinguished element</i>	Elemento distinguido	—
<i>Distributed database</i>	Base de datos distribuida	—
<i>DML</i>	Lenguaje de manipulación de datos	—
<i>Domain declaration</i>	Declaración de dominio	—
<i>Domain</i>	Dominio	—
<i>Ease of integration</i>	Facilidad de integración	—
<i>Editors</i>	Editores	—
<i>Encapsulation</i>	Encapsulación	Encapsulamiento
<i>Entity integrity constrains</i>	Restricciones de integridad de entidades	—
<i>Entity generic</i>	Entidad genérica	—
<i>Entity</i>	Entidad	—
<i>Entity partitioning</i>	Particionamiento de entidades	—
<i>Entity unconnected</i>	Entidad desconectada	—

TÉRMINO ORIGINAL EN INGLÉS	TÉRMINO USADO EN ESTE LIBRO	VOCABLOS ALTERNATIVOS DE USO COMÚN
<i>Entity/relationship model</i>	Modelo entidad/interrelación	Modelo entidad/relación
<i>Equijoin</i>	Equirreunión	—
<i>Exclusive locking</i>	Bloqueo exclusivo	—
<i>Expansion</i>	Expansión	—
<i>Expert system</i>	Sistema experto	—
<i>Expressiveness</i>	Expresividad	—
<i>Extensibility</i>	Extensibilidad	—
<i>External level</i>	Nivel externo	—
<i>External schema</i>	Esquema F	—
<i>Field redefinition</i>	Redefinición de campos	—
<i>File</i>	Archivo	Fichero
<i>First normal form</i>	Primera forma normal	—
<i>Flags</i>	Banderas	Indicadores, señalizadores
<i>Foreign key</i>	Clave ajena	clave externa, llave externa
<i>Form</i>	Formulario	—
<i>Formality</i>	Formalidad	—
<i>Forms analysis</i>	Análisis de formularios	Análisis de formas
<i>Fourth-generation language</i>	Lenguajes de cuarta generación	—
<i>Fourth normal form</i>	Cuarta forma normal	—
<i>Functional schema</i>	Esquema funcional	—
<i>Function schemas</i>	Esquema de funciones	—
<i>Function-driven</i>	Orientado a las funciones	Controlado por funciones
<i>Functional independence</i>	Independencia funcional	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Functional analysis</i>	Análisis funcional	—
<i>Functional model</i>	Modelo funcional	—
<i>Gateway</i>	Compuerta, pasarela	Gateway
<i>Generalization, abstraction</i>	Abstracción de generalización	—
<i>Generalization hierarchies</i>	Jerarquías de generalización	—
<i>Generators and loaders</i>	Generadores y cargadores	—
<i>Global layer</i>	Capa global	—
<i>Global conceptual schema</i>	Esquema conceptual global	—
<i>Global deadlock</i>	Bloque mutuo global	—
<i>Graphic completeness</i>	Compleción gráfica	—
<i>Hierachic database</i>	Base de datos jerárquica	—
<i>Hierarchy inversion heuristic</i>	Heurística de inversión de jerarquía	—
<i>High cohesion</i>	Alta cohesión	—
<i>High-level application</i>	Aplicaciones de alto nivel	—
<i>High overlap</i>	Superposición alta	Solapamiento, recubrimiento alto
<i>Homonym</i>	Homónimos	—
<i>Horizontal partitioning</i>	Partición horizontal	Particionado horizontal
<i>Identifiers</i>	Identificadores	—
<i>Impedance mismatch</i>	Falta de concordancia	Falta de correspondencia
<i>Implementation</i>	Implementación	Implantación, realización
<i>Implicit subsets</i>	Subconjuntos implícitos	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Incremental conceptualization</i>	Conceptualización incremental	—
<i>Index unique</i>	Indice único	—
<i>Indexation</i>	Indexación	—
<i>Infological model</i>	Modelo infológico	—
<i>Inheritance</i>	Herencia	—
<i>Inputs</i>	Entradas	—
<i>Insertion anomaly</i>	Anomalia de inserción	—
<i>Inside-out strategy</i>	Estrategia centrífuga	Estrategia dentro-fuera
<i>Instance</i>	Caso u ocurrencia	Instancia
<i>Integration</i>	Integración	—
<i>Intelligent transformer</i>	Transformadores inteligentes	—
<i>Interactive design</i>	Diseño interactivo	—
<i>Interface</i>	Interfaz	—
<i>Intermediate layer</i>	Capas intermedias	—
<i>Internal level</i>	Nivel interno	—
<i>Interschema properties</i>	Propiedades interesquemáticas	—
<i>Join data-and function-drive</i>	Orientado a los datos y funciones	—
<i>Join</i>	Reunión	Unir
<i>Key constrains</i>	Restricciones de clave	Restricciones de llave
<i>Knowledge</i>	Conocimiento	—
<i>Level of abstraction</i>	Nivel de abstracción	—
<i>Library of designs</i>	Biblioteca de diseños	Librería de diseños

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Link record</i>	Registro de vínculo o enlace	—
<i>Lists</i>	Listas	—
<i>Local layer</i>	Capa local	—
<i>Locking</i>	Bloqueo	—
<i>Log</i>	Bitácora	—
<i>Logical relationship</i>	Interrelación lógica	Relación lógica
<i>Logical child</i>	Hijo lógico	—
<i>Logical model</i>	Modelo lógico	—
<i>Logical link</i>	Vínculo lógico	Enlace lógico
<i>Logical connections</i>	Conexiones lógicas	—
<i>Logical database</i>	Base de datos lógica	—
<i>Loop</i>	Bucle	Lazo, ciclo
<i>Low coupling</i>	Bajo acoplamiento	—
<i>Low overlap</i>	Superposición baja	Solapamiento bajo
<i>Managerial schema</i>	Esquema de gestión	Esquema gerencial
<i>Many-to-many</i>	Muchos a muchos	—
<i>Many-to-one</i>	Muchos a uno	—
<i>Mapping of entities</i>	Correspondencia de entidades	—
<i>Merging of views</i>	Fusión de vistas	Mezcla de vistas
<i>Minimality</i>	Minimalidad	—
<i>Mixed strategy</i>	Estrategia mixta	—
<i>Model network</i>	Modelo de redes	—
<i>Model-independent</i>	Independiente del modelo	—
<i>Model structural</i>	Modelo estructural	—
<i>Model-dependent</i>	Dependiente del modelo	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Modularization criteria</i>	Criterios de modularización	—
<i>Multi-valued</i>	Polivalente	Multivaluado
<i>Multimedia database</i>	Base de datos multimedia	—
<i>Multimember sets</i>	Conjuntos multimiembros	—
<i>n-ary</i>	n-arias	—
<i>Navigation</i>	Navegación	—
<i>Navigation schemas</i>	Esquemas de navegación	—
<i>Neighbour properties</i>	Propiedades vecinas	—
<i>Network database</i>	Base de datos de redes	—
<i>Normal form</i>	Forma normal	—
<i>Null</i>	Nulo	—
<i>Object-oriented</i>	Orientado a objetos	—
<i>Object oriented model</i>	Modelo orientado a objetos	—
<i>Mandatory</i>	Obligatorio	Mandatario
<i>One-to-one</i>	Uno a uno	—
<i>One-to-many</i>	Uno a muchos	—
<i>Ordering</i>	Ordenamiento	Clasificación
<i>Outputs</i>	Salidas	—
<i>Owner-coupled</i>	Acoplado al propietario	—
<i>Parametric text</i>	Texto paramétrico	Texto genérico
<i>Parent pointer</i>	Puntero a hijo virtual	—
<i>Parent-child relationship</i>	Interrelaciones padre-hijo	—
<i>Parent</i>	Padre	—
<i>Partitioning of entities</i>	Partición de entidades	—
<i>Petry nets</i>	Redes de Petry	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Physical child</i>	Hijo físico	—
<i>Physical design</i>	Diseño físico	—
<i>Physical record</i>	Registro físico	—
<i>Physical database</i>	Base de datos física	—
<i>Processing schema</i>	Esquema de procesamiento	Esquema de proceso
<i>Pointer</i>	Puntero	Apuntador
<i>Pointer segment</i>	Segmentos punteros	—
<i>Primary key</i>	Clave primaria	Llave primaria
<i>Primary parent</i>	Padre primario	—
<i>Primary relation</i>	Relación primaria	—
<i>Prime</i>	Primo	—
<i>Process</i>	Proceso	—
<i>Properties</i>	Propiedades	—
<i>Query</i>	Consulta	—
<i>Readability</i>	Legibilidad	—
<i>Record format</i>	Formato de registro	—
<i>Record type</i>	Tipo de registro	—
<i>Records declarations</i>	Declaración de registros	—
<i>Recursive set</i>	Conjunto recursivo	—
<i>Recursive relationships</i>	Interrelaciones recursivas	—
<i>Redundancy</i>	Redundancia	—
<i>Referential integrity constrains</i>	Restricciones de integridad referencial	—
<i>Refinement planes</i>	Planos de refinamiento	—
<i>Refinements</i>	Refinamientos	—
<i>Relation instance</i>	Caso de relación (extensión)	Instancia de relación

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Relational database</i>	Base de datos relacional	—
<i>Relationships</i>	Interrelaciones	Relaciones
<i>Relative search</i>	Búsqueda relativa	—
<i>Repeating groups</i>	Grupos de repetición	—
<i>Repository</i>	Depósito	Repository
<i>Requirements</i>	Requerimientos	Requisitos
<i>Restructuring</i>	Reestructuración	—
<i>Resulting schema</i>	Esquema resultante	—
<i>Reverse engineering</i>	Retroingeniería	Reingeniería, ingeniería inversa
<i>Rings</i>	Anillos	—
<i>Root</i>	Raíz	—
<i>Schema hierarchical</i>	Esquema jerárquico	—
<i>Schema design</i>	Diseño de esquemas	—
<i>Schema network</i>	Esquema de redes	—
<i>Schema balancing</i>	Equilibrio de esquema	—
<i>Schema relational</i>	Esquema relacional	—
<i>Schemas integration</i>	Integración de esquemas	—
<i>Schemas merging</i>	Fusión de esquemas	—
<i>Second normal form</i>	Segunda forma normal	—
<i>Secondary relation</i>	Relación secundaria	—
<i>Select</i>	Selección	—
<i>Self-explanation</i>	Autoexplicación	—
<i>Sets de tuples</i>	Conjuntos de tuplas	—
<i>Shared lock</i>	Bloque compartido	—
<i>Simple attribute</i>	Atributo simple	—
<i>Simple fields</i>	Campo simple	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Simplicity</i>	Simplicidad	—
<i>Single-valued</i>	Monovalente	—
<i>Singular set</i>	Conjuntos singulares	—
<i>Skeleton schema</i>	Esquema armazón	Esquema esqueleto
<i>Support</i>	Apoyar, manejar, soportar	—
<i>Starting schema</i>	Esquema inicial	—
<i>Store field</i>	Campo almacenado	—
<i>Strong entities</i>	Entidades fuertes	—
<i>Structure chars</i>	Diagramas de estructura	Cartas de estructura
<i>Subhierarchy</i>	Subjerarquía	—
<i>Subscripted (repetitive) field</i>	<i>Campo subindicado (repetitivo)</i>	<i>Campo con subíndices</i>
<i>Subset entity</i>	Entidad subconjunto	—
<i>Subsets</i>	Subconjuntos	—
<i>Superset entity</i>	Entidad superconjunto	—
<i>Symbolic pointers</i>	Punteros simbólicos	Apuntadores simbólicos
<i>Table</i>	Tabla	—
<i>Third normal form</i>	Tercera forma normal	—
<i>Tokens</i>	Fichas	Elementos léxicos, componentes léxicos
<i>Tools</i>	Herramientas	—
<i>Tools integrated</i>	Herramientas integradas	—
<i>Top-down primitives</i>	Primitivas descendentes	—
<i>Top-down</i>	Estrategia descendente	Tuple
<i>Tuple</i>	Tupla	—
<i>Twin</i>	Gemelo	—

<i>TÉRMINO ORIGINAL EN INGLÉS</i>	<i>TÉRMINO USADO EN ESTE LIBRO</i>	<i>VOCABLOS ALTERNATIVOS DE USO COMÚN</i>
<i>Update</i>	Actualización	—
<i>Update anomaly</i>	Anomalía de actualización	—
<i>User interface</i>	Interfaz para usuario	—
<i>Vertical partitioning</i>	Partición vertical	—
<i>View</i>	Vista	—
<i>View design</i>	Diseño de vistas	—
<i>Virtual twin pointer</i>	Puntero a gemelo virtual	—
<i>Virtual (or pointer) record</i>	Registro virtual (o puntero) registro	—
<i>Weak entities</i>	Entidades débiles	—
<i>Weak primary relation</i>	Relación primaria débil	—

Índice onomástico

- Abiteboul, S., 61
Abrial, J. R., 59
Aho, A. V., 190, 191
Albano, A., 61, 482, 513
Anwar, T. M., 512, 513
Arora, A. K., 135, 220, 348
Arora, S. K., 392, 427, 464
Atkinson, M., 512, 513
Atre, S., 15
Awong, A. M., 393, 464, 508, 514
- Bachman, C., 427, 491, 513
Bancilhon, F., 513
Barker, R., 96
Barror, O., 255
Batini, C., 60, 62, 96, 135, 155, 219, 513
Batra, D., 62
Beck, H., 135, 512, 513
Beeri, C., 190, 191
Bernstein, P., 61, 190
Bert, M., 348
Bertaina, P., 348, 392, 427
Biskup, J., 156
Bitton, D., 509, 513
Bjoerner, D., 463
Blaha, M., 96
Bolton, D., 219, 438
Bordiga, A., 255
Bostrom, P., 62
Bouzeghoub, M., 134, 483, 513
Bracchi, G., 60
- Braegger, R. P., 485, 513
Braga, A. P., 60
Briand, H., 392
Brodie, M. L., 15, 58, 61, 392
Brown, G., 515
Brown, R., 191
Buneman, P., 156
Burgess, K., 96
- Cardelli, L., 61
Cardenas, A., 14
Carey, M., 511, 513
Carlson, C. R., 220, 348
Carswell, J. L., 276
Casanova, M. A., 60, 425, 449
Catarci, T., 490, 513
Ceri, S., 96, 191, 254, 349, 482, 510, 514
Chen, P. P., 59, 60, 134
Cheng, A., 464
Chikovsky, E., 507, 514
Choobineh, J., 482, 509, 514
Ciardo, C., 348
Civelek, F. N., 156
Codd, E. F., 60, 190, 352, 374, 391
Colletti, R., 96
Colombetti, M., 134
Conklin, J., 509, 514
Constantine, L., 254
Convent, B., 156
Cornelio, A., 276, 515
Cross, J., 507, 514

- Dart, S., 62
Date, C. J., 14
Davis, C., 60
Davis, K. H., 135, 393, 464
Davis, W., 14
Dawson, J., 512, 514
De Antonellis, V., 134
De Marco, T., 254
De Troyer, O., 482, 514
Delobel, C., 464
Demo, B., 135, 348
Dewitt, D., 513
Di Battista, G., 62, 219
Di Leva, A., 135, 348, 392, 427
Dittrich, K., 513
Dogac, A., 156, 482, 514
Dolk, D., 219
Dos Santos, C. S., 59
Doty, K. L., 276
Dou, J., 349
Dudler, A. M., 513
Dumpala, S. R., 392, 427, 464
- Effelsberg, W., 156
Eick, C., 134
Ellis, C. A., 254
Ellison, R., 62
Elmasri, R., 14, 15, 61, 155, 156, 487, 514, 515
Embley, D., 191
Ewusi-Mensah, K., 254
- Fagin, R., 190
Fairley, R., 14
Feifler, P., 62
Ferrara, F. M., 490, 513
Fisher, A. S., 467, 514
Fisher, J. T., 498, 514
Flatten, 96
Flores, B., 134
Freedman, P., 220
Friedel, M., 515
Fry, J., 15, 96, 348, 392, 481, 515
Furtado, A. L., 59
- Gadgil, S. G., 155
Gala, S. K., 157
Gane, C., 14, 254, 466, 514
Gardarin, G., 134, 513
Giolito, P., 348, 392, 427
Goldberg, 511, 514
Goldstein, R., 62, 487, 515
Goodman, N., 190
Gottlob, G., 191
Greenspan, S., 255
Gualandi, P. M., 60
Guida, G., 134
- Habermann, N., 62
Habrias, H., 392
Hagelstein, J., 60
Hammer, M., 60
Hazzah, A., 499, 514
Hevner, A., 514
Hoffer, J. A., 14, 62
Howe, D. R., 15
Hue, J. F., 392
Huffman, S., 219
Hull, R., 61
- Jacobelli, C., 348
IEEE, 509
Inmon, W. H., 15
Irani, K., 428
- Ji, W., 220, 348
Johannesson, E., 393
Johnston, T., 393
- Kalman, K., 393
Kangassalo, H., 60, 219, 482, 507, 514
Kappel, G., 61
Katz, R., 392, 428
Kent, W., 15, 59, 191
Kerschberg, L., 59, 219
King, R., 61

- Klug, A., 59
 Knapp, D., 463
 Koenig, J., 219, 348
 Korth, H., 14
 Kunii, T., 15
- Larson, J., 156, 487, 514, 515
 Lazimy, R., 61
 Leben, J. F., 463
 Lehman, J., 515
 Lenzerini, M., 155, 513
 Lien, E., 464
 Ling, T., 191
 Liu, 60
 Lochovsky, F., 59, 60, 463
 Lockemann, P. C., 134
 Loomis, M. E., 15
 Lovengen, H., 463
 Lum, V., 15
 Lundberg, M., 96, 254
- MacDonald, G., 96
 Maier, D., 471, 513, 514
 Mannino, M., 156
 March, S., 60
 Marcus, H., 157
 Markowitz, V. M., 393
 Marrone, V., 348
 Martin, J., 14
 Maryansky, F., 61
 Mattos, N. M., 96
 McClure, C., 467, 514
 McCubrey, D., 96
 McFadden, F. R., 14
 McGee, W., 463
 McKee, R., 515
 McLeod, D., 60
 Meersman, R., 482, 514
 Melkanoff, M. A., 191
 Metais, E., 513
 Meyer, B., 511, 514
 Michels, M., 96
 Miller, D., 220
- Millman, J., 509, 513
 Motro, A., 156
 Myers, G. J., 218
 Mylopoulos, J., 61, 255, 482, 514
- Navathe, S. B., 14, 15, 60, 61, 135, 155, 156, 157, 219, 276, 349, 393, 428, 464, 487, 508, 510-511, 513, 514, 515
 Neuhold, E. G., 59
 Nilsson, E. G., 135
- O'Riordan, P., 96
 Olle, T. W., 96, 276
 Orsini, R., 61
- Pace, B., 494, 515
 Palmer, I. R., 96
 Paolini, P., 60
 Parker, D. S., 191
 Peckham, J., 61
 Pelagatti, G., 60
 Peterson, J. R., 254
 Pillallamarri, M. K., 61
 Prabandham, M., 499, 515
 Premerloni, W., 96
 Proix, C., 134
 Purkayastha, S., 428
- Quang, P. T., 494, 515
- Rambaugh, J., 96
 Rebsamen, 513
 Reiner, D., 468, 470, 477, 482, 515
 Rheingans, P., 515
 Ridjanovic, D., 59
 Rifaut, A., 60
 Robson, D., 511, 514
 Rochfeld, A., 96
 Rolland, C., 134
 Rosenthal, A., 470, 482, 515
 Ross, D., 96, 254

- Sakai, H., 348, 464
Santucci, G., 96, 219
Sarson, T., 14, 254
Savasere, A., 157
Scheuermann, P., 59
Schiffner, G., 59
Schkolnick, M., 60, 348
Schmidt, J., 61, 392
Schrefl, M., 61
Sheth, A., 157, 487, 515
Shipman, D. W., 61
Shoman, K., 96, 254
Shoshani, A., 393
Silberschatz, A., 14
Simon, Y., 392
Smith, D. C., 59
Smith, J. M., 59
SOFTECH, Inc., 254
Sol, H. G., 96, 276
Somalvico, M., 134
Sorensen, P., 348
Sowa, J. F., 134
Spaccapietra, S., 60, 156
Storey, V., 62, 487, 515
Su, S. Y. W., 58, 62
Swenson, J. R., 392
- Tamassia, R., 219
Teorey, T. J., 15, 96, 219, 348, 392, 393,
 428, 481, 483, 510, 515
Torgersen, S., 509, 513
Tsichritzis, D. C., 59, 463
- Tucherman, L., 60
Tully, J., 276
- Ullman, J., 14, 15, 190, 191, 471, 514
- Vardi, M., 471, 514
Verlinden, P., 482, 514
Verrijn-Stuart, A. A., 96, 276
- Weber, H., 59
Weeldryer, J., 514
Wei, G., 219, 348
Weldon, J. L., 15
Wiederhold, G., 15, 61, 155, 349
Williams, J. L., 494, 515
Williams, S., 427
Winans, J., 464
Winkler, J., 219
Wong, H. K., 61, 392, 428
- Yang, D., 96, 392, 393, 481, 515
Yao, S. B., 15
Yourdon, E., 254
- Zaniolo, C., 191
Zodnik, S., 513
Zehnder, C. A., 513
Zilles, S. N., 15
Zoeller, R., 219

Índice analítico

- abstracción, 17
- agregación, 19
- clasificación, 18
- generalización, 20, 28-30
- mecanismos de, 1
- acoplamiento, 195
- actograma, 248
- actividades
 - (en redes Petri), 250
 - típicas del diseño conceptual, 9
- ADABAS, 491, 496
- AD/CYCLE, 473, 488, 494, 497, 498, 510
- administrador de bases de datos, 470, 493
- agregación, 22
 - binaria, 22
 - de muchos a muchos, 25
 - de muchos a uno, 25
 - de uno a muchos, 25
 - de uno a uno, 25
 - n-arias, 22, 25
- agrupamiento, 331
- alias (véase también *sinónimo*), 479
- almacén de datos, 225
- análisis
 - de conflictos, 139, 142
 - estructurales, 147
 - de datos, 247, 257
 - de formularios, 110
 - de requerimientos, 92
 - de sistemas, 229
 - estructurado, 494
 - final, 262
- funcional, 9-12, 221, 257, 472
- textual de los requerimientos, 100-101
- analizador, 469, 479
- anomalía, 174-175, 331
 - de actualización, 174-175, 331
 - de eliminación, 175
 - de inserción, 175
 - en la modificación (véase *anomalía*)
- anotación (en diseños), 474
- ANSI (Instituto Nacional Americano de Estándares), 33
- ANSI/SPARC proposición, 33
- aplicación
 - carga de la, 316
 - diseño de aplicaciones de alto nivel, 9-11
 - diseño de programas de, 9-11
 - especificaciones de, 9
 - programa de, 221
 - requerimientos de aplicaciones, 9 de alto nivel, 9
- árboles de ocurrencias jerárquicos, 432
- archivo, 117
- arquitectura cliente/servidor, 510
- ASK Sistema de Computadoras, 351
- aspectos procedimentales, 238
- atributo(s), 39, 89-90, 352, 396
 - asociados (véase *atributos*)
 - compuestos, 43, 358, 402, 442
 - compuestos y polivalentes, 402, 442-443
 - derivados, 169, 317
 - monovalente (atributo), 39

- no nulos, 483, 487
- pertinentes, 39, 176
- polivalentes y repetidos, 39-40, 397, 402, 443, 481, 488
- simple, 40, 89-90
- autoexplicación, 2, 163
- autonomía local, 510

- BACHMAN/DBA, 492
- balanceo de esquemas, 200
- bandera, 123
- base de conocimientos, 97, 209
- base de datos, 4
 - multimedia, 209
 - distribuidas, 332
 - orientadas a objetos, 61-62, 427
- biblioteca de esquemas, 466

- caja de herramientas, 491
- calidad de un esquema de base de datos, 2, 160, 194
- campos, 120, 438
 - simples, 120
 - subindicados (repetitivos), 120
 - tipo de, 430
- capa (de un diccionario), 211-212, 215
 - global, 212
 - intermedia, 212
 - local, 212
- cardinalidad, 25, 474
 - de una relación, 352
 - máxima, 24
 - máxima (de clases), 24, 26
 - mínima, 23
 - mínima (de clases), 23, 25
 - promedio, 313
- carga de bases de datos, 313-317
- carga y generación del esquema, 470, 483
- casos válidos (de un esquema), 32
- CDD/Plus, 473, 494, 499
- ciclo de vida, 5, 466
- ciclos de interrelaciones, 167

- clase, 18
 - genérica, 28
 - subconjuntos, 28
- clave, 483, 495
 - ajena, 355, 488, 491, 492, 495
 - candidata, 353, 480, 488, 491
 - jerárquica, 444
 - primaria, 332-334, 353, 481, 492, 507
 - restrictión de, 354
 - única indizada, 332
- Clipper, 491
- co-conjunto, 398
- códigos de comandos (véase *IMS*)
- cohesión, 195, 220
- compleción, 2, 160, 241
 - (de un esquema), 160, 474
 - (de un esquema funcional), 241
 - (de una primitiva), 74
 - mutua, 260
- compresión (de requerimientos), 204
- compuerta, 510
- Computer Associates (compañía), 395, 419
- Computer Corporation of America, 470
- Computer Systems Advisors (compañía), 494
- concepto de actualidad
 - en el modelo de redes, 412
 - en el modelo jerárquico, 449
- concepto o conceptos
 - balanceo de, 200
 - compatibles, 147
 - desequilibrio de, 200
 - discrepancia de, 144
 - idénticos, 147
 - incompatibles, 147
 - propiedad de un, 144
 - restrictiones de, 145
 - semejanza de, 144
 - vecinos, 144
- conceptualización inicial, 92
- condiciones previas y posteriores, 250, 253
- conexión lógica, 66
- conflictos
 - de nombres, 142
 - estructurales, 142

- conjunto(s), 249
acoplado al propietario, 398
de mensajes, 249
elemento distinguido, 397
mixtos, 249
multimiembros, 401, 420-421
ocurrencia del, 396
orden del, 398
reales, 249
recursivo, 400
singulares, 401
tipo de registro miembro, 398
tipo de registro propietario, 398
tipos de, 396
CONNECT (modelo red), 412
contenido de información, 165
convenciones de nombres, 471, 474, 488, 493
correcciones de un esquema conceptual, 2, 160
semánticamente correcto, 160
sintácticamente correcto, 160
correcciones de un esquema funcional, 241
correspondencia de esquemas
del modelo ER al modelo relacional, 356-368
del modelo ER al modelo de redes, 401-411
del modelo ER al modelo jerárquico, 437-447
correspondencia binaria, 23
creación de prototipos, 6, 491, 496
criterios de modularización, 195
- data Manager** (producto), 497
DATAID, 482
datos de intersección, 435
datos derivados, 116, 481, 507
DB2, 332, 473, 491, 492, 494, 497, 498
dBASE, 473, 491
DBMS (véase *gestor de bases de datos*)
DDEW, 483, 487
declaraciones de dominio, 40
declaraciones de registros, 99
- dependencias
de inclusión, 365, 378, 483, 484
funcionales, 173, 332, 377, 480, 483, 484
multivaluadas, 191
depósito, 210, 472, 489, 494, 498-499
descripción textual, 1
desnormalización, 331, 507, 510
DFD
equilibrar el, 242
integración, 235
diagrama
Arthur Young, 497
Bachman (diagrama de modelo de datos), 494
Chen (ER diagram), 494
Constantine, 494
de Bachman, 397
de trabajo, 491
de descomposición, 469
de estructura, 469
DeMarco, 469, 497
editor de, 476-477, 483, 491
espacio de pantalla, 467-469
Gane/Sarson, 469, 494, 497
Jackson (diagrama de estructura), 494
Martin, 469, 497
Merise (diagrama ER), 494
Yourdon (flujo de datos), 469
diccionario de datos, 209, 243, 466, 470, 479, 494-499
activos (diccionarios), 211
diccionarios locales, 215
diseño del, 213
estructura de un, 211
estructura recursiva para el diccionario, 212
integración de, 215
sistemas de, 33
diccionario de recursos de información (IRDS), 35, 219, 499
Digital Equipment Corporation, 395, 419, 473, 494, 499, 510
DISCONNECT (modelo redes), 412
diseño, 6

- asistido por computador (CAD), 512
- biblioteca de, 466, 475
- comprobación de entrada/salida, 472
- conceptual, 4, 7
- de bases de conocimientos, 97
- de bases de datos, 1
- de bases de datos conceptual, 1
- de vistas, 99, 470, 472, 475, 480, 481, 505, 508
- enfoque conjunto orientado a los datos y funciones, 9
- físico, 2, 8, 312, 483, 491-493, 494, 507, 509
- inicial, 261
- orientado a las funciones, 9, 262, 496, 505
- orientado a los datos, 262, 495, 505
- proceso interactivo, 471
- rendimiento de un, 469, 471, 492, 505, 508-509
- seguimiento de la metodología y el, 465, 467, 472, 505
- diseño lógico, 2, 7, 309-313
 - de alto nivel, 307
 - dependiente del modelo, 307, 310-312, 361
 - entradas al, 310-312
 - independiente del modelo, 307, 309-312
 - modelo de, 30
- DL/1, 447
- DML
 - de red, 411-414
 - jerárquico, 447-450
- documentación (véase *esquema de documentación*)
 - y mantenimiento de esquemas, 193, 203, 213
- documentos de diseño, 92
- dominio, 40, 484, 493
- DOS, 509
- editor (en herramientas de diseño), 469
- Eiffel, 511
- elementos de datos, 396, 430
- eliminación de conjuntos, 399
- encapsulación, 511
- encyclopedia, 466, 497
- entidad(es), 36
 - débiles, 46
 - desconectadas, 479
 - fuertes, 46, 492
 - genérica, 43, 320
 - principal, 43
 - subconjunto, 43
 - superentidad, 320
- equirreunión, 353
- equivalencia
 - en las fronteras, 241
 - entre las construcciones del modelo, 138
- ERASE (modelo redes), 412
- ERwin (herramientas), 488, 491
- espacio de pantalla, 467
- especialización, 41
- especificación de requerimientos, 6
- especificaciones de diseño incompatibles, 139
- esquema
 - armazón, 1, 85, 141, 194
 - conceptual, 1, 7
 - D, 257
 - de datos, 90, 469, 475, 480
 - de operación, 271
 - de procesamiento, 223
 - externo de proceso, 258
 - F, 257
 - generador, 470, 483
 - jerárquico (véase *esquema*)
- esquema de bases de datos
 - de redes, 401
 - jerárquico, 430-432
- esquema(s), 30
 - conceptual, 312-313
 - conceptual a lógico, 7, 92, 312, 344
 - conocimiento extensivo, 28
 - conocimiento intensivo, 32
 - corrección de, 160
 - de funciones, 9, 92, 223, 469, 475-477, 480

-
- de navegación, 222, 271, 315
 - descendente, 142
 - descubrir redundancia de un, 471, 490
 - equivalentes, 165
 - externos, 257-258
 - final, 141
 - físico, 8
 - global, 142, 194, 213-214
 - inicial, 141
 - integrado, 140
 - intermedios, 141
 - legibilidad, 161
 - locales, 214
 - lógico, 7
 - metaesquema, 431
 - minimal, 160, 168
 - resultante, 66, 227
 - se explica a sí mismo, 163
 - estadísticas
 - agregaciones, 62
 - aplicaciones, 62
 - estados (lugares) en redes Petry, 250
 - de entrada, 250
 - de salida, 250
 - estrategia, 67
 - ascendente, 80, 230, 470, 475
 - centrífuga, 82, 233, 470, 475, 508
 - descendente, 78, 470, 475
 - hacia atrás, 233
 - hacia delante, 233
 - mixta, 84, 234, 470, 475, 508
 - orientada a la salida, 233
 - estrategias de diseño de un esquema
 - ascendente, 80-81
 - centrífuga, 82-84
 - descendente, 78-80
 - mixta, 84-85
 - estrategias para el diseño
 - ascendente, 80
 - centrífuga, 82
 - descendente, 78
 - mixta, 84
 - estructura de la oración inglesa, 134
 - estructura de un diccionario de datos, 212
 - integración de diccionario de datos, 212
 - las vistas, 212
 - estudio de factibilidad, 5
 - Excelerator, 468, 488, 494, 499, 510
 - EXODUS (sistema), 511
 - expansión (de requerimientos), 204
 - expresividad
 - de un esquema, 161
 - de un esquema conceptual, 2, 474
 - del modelo conceptual, 34
 - extensibilidad, 2, 164, 468, 512
 - extensión de una relación, 352
 - fabricación asistida por computador (CAM), 512
 - facilidad de integración, 240
 - facilidad de lectura, 35
 - fichas, 250
 - FIND (modelo redes), 412
 - flexibilidad, 240
 - flujo (véase *flujo de información*)
 - flujo de datos, 225
 - diagrama de, 224
 - editor de diagramas de, 477
 - flujo de información
 - física, 248
 - papel (para flujos), 248
 - FOCUS (sistema), 496, 497
 - forma normal, 164, 190, 356
 - cuarta, 191
 - de Boyce Cod, 165, 182, 374, 377
 - primera, 164, 377
 - segunda, 178, 332, 377
 - tercera, 179, 332, 374, 377
 - formalidad, 34
 - formatos de registros, 99
 - formulario, 91
 - FORTRAN, 117
 - fragmentos (de interrelaciones), 510
 - FROM cláusula (en SQL), 369
 - función, 221, 491
 - fusión
 - de entidades e interrelaciones, 331-332
 - de esquemas, 140

- fusión de vistas, 148, 471
 enfoque de pares, 487
 integración n-aria, 487
- GALILEO, 61
 GAMBIT, 485
 generalización jerárquica, 41, 436, 441-442, 479-481, 485-486, 492
 generalizaciones superpuestas o parciales, 322
 GESI, 489
GET
 en modelo de redes, 447-450
 glosario de conceptos de un esquema funcional, 226
 glosario (de términos), 103, 226
 de un esquema funcional, 226
 grado de una relación, 352
 gráficas de actividad ISAC, 248, 254
 gramática BNF, 48
GROUP BY (en SQL), 369
 grupos afines, 483
- HAVING** (en SQL), 369
 herencia, 21, 43, 511
 herramienta (véase *herramientas de diseño*)
 herramientas de diseño, 466
 árbol de diseño, 483
 arquitectura de referencia, 465, 475
 automatizadas, 465-466, 506, 508
 comerciales, 488, 499-505
 conjunto de, 469, 494, 508
 heurísticas, 470, 480, 509
 robustas e integradas, 465, 471, 508
 herramientas heurísticas, 470
 heurística de inversión jerárquica, 438, 450
 Hewlett Packard, 395
 hojas electrónicas de cálculo, 209
hold option (GET in IMS), 449
 homónimo, 102, 144
 Honeywell, 395, 419, 487
 Hypertext, 509
- IBM, 351, 473, 482, 494, 497-499
 ICN (véase *control de información de red*)
IDEF1-X, 491-492
 identificador(es), 44
 externo, 46
 eliminación de, 356, 404, 437
 interno, 46, 357
 mixto, 46
 múltiple, 334
 simple, 46
IDMS, 491
IFO, 61
IMAGE (sistema), 395
 implementación, 6
IMS (sistema gestor de información), 433, 482, 491, 507
 base de datos física, 433, 442
 base de datos lógica, 445-447
 campo, 438
 códigos de mandato D*, 448
 hijo físico, 437
 hijo lógico, 437
 interrelación física padre-hijo, 443
 interrelación padre-hijo real, 437
 interrelaciones lógicas, 439
 registro físico, 433
 segmentos punteros, 436
 independencia funcional, 229
 índice, 469, 483, 490, 493, 494
 único, 334
 Infodyne, 489
 Informix, 491
 ingeniería de software asistida por computador (CASE), 465, 467, 473, 494-499, 502-504, 505
 enfoque orientado a los datos, 466
 herramienta para, 466
 inserción de conjuntos, 398
 instancia de un esquema, 32
 integración (véase *integración de vistas*)
 integración de bases de datos, 137
 integración de esquemas (véase *integración de esquemas*)
 herramientas para la, 487
 integración de vistas, 2, 93, 137, 151, 212

- integridad de entidades (véase *restricciones*)
integridad referencial (véase *Restricción*)
IntelliBase, 482
interfaces gráficas de usuario (véase *interfaz de usuario*)
interfaz (véase *interfaz de usuario*)
interfaz de usuario, 225, 465, 466, 482, 506
 habitacional, 467
interrelación(es), 36, 325
 anillos, 36-37
 binarias, 36, 409
 ciclos de, 167
 correspondencia de operaciones del ER
 al modelo de redes, 411-415
 de muchos a muchos, 331-332, 365,
 408, 432, 433, 438, 482
 muchas a una, 32
 n-arias, 36, 366, 409, 438
 padre-hijo (véase *interrelación padre-hijo*)
 partición de (véase *partición*)
 postula, 480
 recursivas (véase *Recursivas*)
 subconjunto, 382
 transformación de esquemas ER a jerárquicos, 438
 uno a muchos, 37, 331-332, 363, 408,
 420, 432, 438, 482
 uno a uno, 331-332, 361, 405, 438
interrelación padre-hijo, 430
 tipos de, 430
 virtuales, 434, 445-447
- jerárquico, 430
- knowledge Ware, 497
- legibilidad, 2, 242, 470
 conceptual y gráfica, 242
 de un esquema, 161
- lenguaje C, 117
- lenguaje C objetivo C, 511
lenguaje C++, 511
lenguaje de cuarta generación, 470, 497
lenguaje de definición de datos, 8, 470, 491-493, 494-497
lenguaje natural, 91
 descripciones en, 99
 requerimientos expresados en, 100-101
- lista, 115
- Logic Works, Inc., 491
- LOTUS, 473
- lugares (véase *estados*)
- Macintosh, 509
- mantenimiento de esquemas, 2, 203, 242
 MastER Plus, 488-491
- mesa de trabajo de ingeniería de información (IEW), 480, 497
- metarreglas, 485
- metodología, 65, 90, 96, 100, 110, 141, 194,
 238, 261, 271, 276, 470, 472, 505
 automatizar una, 473
 flexibilidad metodológica, 473
 neutral, 470, 497
 para el análisis funcional, 238
- metodologías de diseño, 1
- métodos, 511
- minimalidad, 2, 34, 242
 de un esquema, 160
 de un esquema funcional, 242
 de una primitiva, 74
- modelo 204 (dbms), 470, 497
- modelo conceptual (de datos), 7, 30
- modelo de cascada, 466
- modelo de datos, 7, 30
 binarios, 11
 conceptual, 7, 30
 de redes, 7, 396-401, 483
 estructural, 11
 funcional, 11
 jerárquico, 7, 430-437
 lógicos, 7, 30, 483
 relacional, 7, 352-356, 481-482, 485
 semántico, 11

- modelo de datos relacional (véase *modelo de datos*)
modelo de entidades-interrelaciones, 1, 11, 474, 483, 492, 494, 510
modelo entidad-categoría-interrelación, 156, 487
modelo funcional, 247
modelo infológico, 59
modelo jerárquico de datos (véase *modelo de datos*)
modelo orientado a objetos, 61, 97, 255, 276, 419, 423
modelo(s)
modelo(s) conceptuales, 30
modelo(s) de datos, 30
modelos de funciones, 9
modelo(s) lógicos, 30
MODIFY (modelo de redes), 412
modularización, 194
MODULA/R (lenguaje), 485
Motif, 509
- NDL (lenguaje de definición de redes), 411
NeXT (máquina), 509
NeXTStep (herramienta), 509
NIAM, 510
nivel de abstracción, 209-210
niveles (planos) de refinamiento, 194, 200, 201, 215
Nomad Focus, 497
nombres de conceptos, 66, 225
normalidad, 2, 164
normalización, 469, 471, 480, 483-485, 488, 491-492, 494, 495, 505, 507
- O2 (dbms), 511
Object/1, 511
OCCURS (cláusula de COBOL), 121-122
Ontos, 511
opciones de modalidad o implantación, 399
Open Software Foundation, 509
operación, 6, 221
operación de bases de datos, 270
- optimización de consultas, 312
ORACLE, 351, 473, 491, 497
orden
 del conjunto, 349
 del conjunto, 398
organización, 210, 220, 242
OS/2, 509
- panel posterior de software, 498
paradigma estructura función (F-S), 276
parallelismo, 332
partición
 de entidades, 327-330, 332
 de interrelaciones, 330
 de un subesquema, 194
 horizontal, 327-330
 vertical, 327-330
partición horizontal (ver *partición*)
partición vertical (véase *partición*)
participación
 obligatoria, 24
 opcional, 24
patrones de refinamiento, 270
persistencia (de objetos), 511
personificación de la interfaz de usuario, 494, 506, 510
perspectivas diferentes (en integración de vistas), 138
PL/1, 117
políticas, 499, 508, 510
POSE, 488, 495
primitivas
 ascendentes, 72, 227
 descendentes, 67, 68, 227
 para el análisis funcional, 227, 267
 refinamientos, 200, 267
 transformación de esquemas, 67
procesos, 224
proingeniería, 492
PROLOG, 191, 485, 487
propagación de cambios, 465, 485
propiedad (de un concepto), 144

-
- propiedades de cobertura de la generalización, 28-30
 cobertura total o parcial, 28
 cobertura exclusiva o superpuesta, 28-30
 propiedades interesquemáticas, 145
 puntero
 a gemelo virtual, 436
 a padre virtual, 436
 a hijo virtual, 436
 punteros simbólicos, 122
 puntos de entrada, 401
- rasgos procedimentales, 239
 RBase, 491, 497
 recolección y análisis de requerimientos, 5-6, 483
 RECONNECT, 412
 recorrido en preorden (de un árbol), 433
 recursividad, 239
 recursos de importación y exportación, 470, 473, 474, 497
 redefinición de campos, 122
 redes de control de información, 250
 redes Petry, 250, 254
 refinamiento(s), 67, 194-212, 260
 ascendentes, 106
 centrífugos, 106
 descendentes, 106
 procedimental, 239
 típicos sugeridos, 270
 registro o vínculo o enlace, 408, 409, 415
 registros, 117, 396, 430
 acceso a registros lógicos, 481, 483
 relación, 352
 de una tupla, 352
 primaria, 380
 primaria débil, 380
 secundaria, 380
 rendimiento (véase *diseño*)
 representación(es), 35
 gráficas, 35
 lingüísticas, 35
 reproducidos, 511
 requerimientos
 de aplicaciones, 91
 de datos, 91
 de funciones, 91
 resolución (de conflictos), 140
 restricción estructural, 405
 restricciones, 144
 de clave, 354
 de integridad de entidades, 355
 de integridad referencial, 355, 491, 494
 restricciones de integridad (véase *restricciones*)
 restructuración (véase *reestructuración de esquema*)
 restructuración de esquemas, 2, 185
 retroingeniería, 212, 466, 488, 491, 494, 497, 507
 de esquemas de redes a esquemas ER, 418-422
 de esquemas jerárquicos a esquemas ER, 455-458
 de los esquemas relacionales a esquemas ER, 374-387
 reunión, 353
 RIDL*, 482
- SABRINA, 484
 SADT, 247
 SAS Inc., 429
 SECSI (herramienta), 482, 484
 selección de conjunto, 398
 SELECT (en SQL), 369
 separabilidad, 240
 simplicidad, 34
 sincronización de procesos, 247
 síntesis de esquemas, 480
 sistema abierto, 472
 sistema de automatización de oficinas, 247
 sistema de bases de datos distribuidas, 137, 510-511
 sistema de bases de datos relacional, 2, 351
 sistema de creación de vistas (VCS), 487
 sistema de información, 4
 sistemas de gestión de bases de datos, 4

sistemas de transformación, 253
microtransformadores, 470
transformadores de mantenimiento del contenido, 469
transformadores, 469
transformadores inteligentes, 470
sistemas dependientes del tiempo, 247
sistemas expertos, 134, 156, 483, 487, 491-493, 497, 506
Smalltrack (lenguaje), 511
SQL, 351, 369, 494, 497
SQL/DS, 496
STORE (modelo de redes), 412
subconjunto
 de entidades (véase *subentidades*)
 implícito, 169
subentidad, 171, 320-327
 colgante, 171
subesquema, 2
superentidad, 320-327
superposición, 204
supuesto de la relación universal, 471
SYSbase, 352
System 2000, 429, 439, 443
System Application Architecture (SAA), 498

tabla, 115
 de frecuencia de operaciones, 316
 de volumen de acceso de operaciones, 316
TAXIS, 61, 482
teoría relacional, 2
texto paramétrico, 111
tiempo, 255
tipo de objeto de interacción, 276
tipo de registros, 117, 396, 430
 hijo, 430
 miembro, 398-400

padre, 430
propietario, 397-400
raíz, 431
virtual o puntero, 434
trabajo gráfico flexible (véase *diagramas*)
transacción, 469, 480, 494
transformación (véase *esquema de transformación*)
transformación(es) de esquemas, 66, 165, 194, 227, 481-482, 490-491
 de aumento, 166
 de reducción, 166
parcialmente aplicables, 206
retiradas, 205
 sin cambios, 205
transición, 250
transparencia de localización, 510

unidades de organización, 210
Univac, 395, 419
Unix, 484
User Work Area, 447

validación y prueba, 6
valor nulo, 355-356
vectores, 397
vínculo (enlace) lógico, 227
vista, 2, 212, 459, 494
vista de esquemas jerárquicos, 445
vista lógica, 445-446
volumen de datos, 313
 tabla de datos, 315
VSAM, 492

WHERE (en SQL), 369
Windows, 491, 509.

Diseño conceptual de bases de datos

UN ENFOQUE DE ENTIDADES-INTERRELACIONES

Carlo Batini, Stefano Ceri y Shamkant B. Navathe

Este libro se ha escrito para cubrir las necesidades de los diseñadores, programadores y usuarios finales de bases de datos, interesados en maximizar la potencia del diseño lógico y conceptual, utilizando el enfoque entidad-interrelación (E/R). Los autores, expertos internacionalmente reconocidos, examinan en profundidad el diseño conceptual, el análisis funcional y el diseño lógico con énfasis en las cuestiones relativas al usuario y a la aplicación. El texto presenta una metodología de diseño paso a paso única que incorpora los últimos avances de la ingeniería de software y las técnicas de diseño de bases de datos. Un capítulo sobresaliente, escrito por el Dr. David Reiner, proporciona una revisión de herramientas de diseño de bases de datos que incluye las herramientas fundamentales para el diseño lógico y conceptual, y el diseño de bases de datos reales y herramientas CASE.

Características

- Enseña cómo conceptualizar, diseñar y hacer ingeniería inversa o modificar grandes sistemas de bases de datos relacionales, jerárquicas o de red.
- Introduce y explica conceptos y sus implementaciones utilizando una metodología paso a paso que es accesible a los diseñadores de bases de datos, usuarios finales y programadores de aplicaciones.
- Enfatiza completamente el modelo entidad-interrelación, abarcando jerarquías de generalización, jerarquías conjunto-subconjunto y una diversidad de restricciones semánticas.
- Integra el análisis funcional con el análisis de datos para mostrar cómo crear un entorno integrado de bases de datos para una diversidad de aplicaciones.
- Examina la correspondencia o concordancia inversa en el esquema E/R para los modelos relacional, de red y jerárquico.
- Ilustra cómo implementar técnicas de diseño mediante una aplicación de un caso práctico, amplio y real.
- Proporciona una revisión actualizada y el análisis de herramientas de diseño de bases de datos.



ADDISON-WESLEY IBEROAMERICANA, S.A.
7 Jacob Way, Reading, Massachusetts
01867 U.S.A.

EDICIONES DÍAZ DE SANTOS, S.A.
Juan Bravo, 3-A
28006 Madrid, España

9 780201 601206

ISBN 0-201-60120-6