UNIVERSIDAD DE VIC

MÁSTER EN TRADUCCIÓN ESPECIALIZADA

TRADUCCIÓN TÉCNICA DEL CAPÍTULO III DEL LIBRO *ADVANCED DATA WAREHOUSE DESIGN: FROM CONVENTIONAL TO SPATIAL AND TEMPORAL APPLICATIONS*

TESINA POR PATRICIA M. BRENES ROLDÁN

TUTORA: CARME MANGIRON

SEPTIEMBRE DE 2011

AGRADECIMIENTOS

No hubiera podido finalizar este trabajo sin la valiosa ayuda de mi tutora Carme Mangiron, quien mostró gran interés durante todo el proceso de traducción y preparación del análisis. Fue realmente valioso contar con su disponibilidad continua y su respuesta rápida a todas mis preguntas, así como con sus sugerencias y consejos.

Asimismo, deseo agradecer a la Dra. Elzbieta Malinowski, por su apoyo constante durante el proceso de traducción y su paciencia durante mi proceso de aprendizaje acerca del tema de almacenes de datos.

Por último, mi agradecimiento va también a Jordi Chumillas, Director del Máster, quien durante los dos años de carrera también me brindó su apoyo en los momentos difíciles.

A mi jefa, Claudia Engle, quien encabeza la Sección de Traducciones del Banco Interamericano de Desarrollo (BID), porque siempre ha apoyado mi superación profesional y fue uno de los motores que me empujó hasta el final. En general, a todos mis colegas que me acompañaron durante todo el proceso.

A todos los miembros de mi familia y a mis amigos y amigas, por ser mis porristas favoritos y tener para mí siempre palabras de ánimo que me ayudaron a seguir adelante desde el primer día que empecé la carrera.

A mis compañeros y profesores del Máster con los cuales establecí además una hermosa amistad.

Gracias a todos y todas.

ÍNDICE

1 INTRODUCCIÓN 1
1.1 Antecedentes 1
1.2 Objetivos 2
1.3 Metodología 2
1.3.1 Búsqueda de información 2
1.3.2 Traducción del texto 4
1.3.3 Traducción de figuras 5
2 ALMACENES DE DATOS 6
2.1 Historia de los almacenes de datos 6
2.2 Definición 7
3 TRADUCCIÓN TÉCNICA 10
3.1 Importancia y definición 10
3.2 El papel del traductor 12
3.3 Teoría de la traducción técnica 13
3.4 Teoría de Skopos 14
4 TERMINOLOGÍA 16
4.1 Definición y diferenciación de términos 16
4.2 Terminología y su normalización (ISO) 18
4.3 Terminología y ciencia de la información 20
5 PROCESO DE TRADUCCIÓN 22
5.1 Objetivo 22
5.2 Glosario 22
5.3 Recopilación de la terminología 25
5.3.1 Almacén, bodega o depósito 26
5.3.2 Explicación de términos 27
5.3.3 Significados dobles 27
5.3.4 Uniformidad 29
5.3.5 Uso de sinónimos 29
5.3.6 Regionalismos 30
5.3.7 Limitaciones de la Internet para la búsqueda de términos 31

5.3.8 Errores en el original 32
6 ESTILO 33
6.1.1 Lengua materna de los autores de libro 33
6.1.2 Uso de cursiva versus negrita 34
6.1.3 Fusión de oraciones 34
6.1.4 Repeticiones 35
6.1.5 Puntuación 35
6.1.6 Omisiones 36
6.1.7 Adiciones 37
6.1.8 Cacofonía 38
6.1.9 Orden de oraciones 38
7 CONCLUSIONES 38
8 BIBLIOGRAFÍA 42

ANEXOS[1]

Anexo 1 Glosario……..…….51
Anexo 2 Glossary……..…….....62
Anexo 3 Nuevos términos propuestos……..…….……75
Anexo 4 Capítulo III "Almacenes de datos convencionales"…….….…77
Anexo 5 Chapter III "Conventional Data Warehouses"……..……...135

[1] Los anexos se presentan en el orden español/inglés debido a dificultades técnicas a la hora de unir el capítulo 3 en inglés que está en formato pdf al documento principal que está en formato Word.

1 Introducción

1.1 Antecedentes

La propuesta de realizar la presente tesina surgió a partir de conversaciones con la Dra. Elzbieta Malinowski Gajda, quien en el año 2008 publicó, junto con el Dr. Esteban Zimányi, el libro titulado *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications (Data-Centric Systems and Applications)*. El trabajo se realizó bajo la supervisión y constante retroalimentación de la Dra. Malinowski. Vale aclarar que el Dr. Zimányi no tuvo participación en este proyecto puesto que él labora en la Universidad Libre de Bruselas y la traducción fue elaborada para el beneficio directo de la Universidad de Costa Rica (UCR), donde la Dra. Malinowski es catedrática en la carrera de Computación y Ciencias de la Información.

Vale la pena mencionar la importancia y gran influencia de este libro en su género, según se puede constatar por la siguiente crítica realizada por William Inmon (2009):

> This book is a great achievement as it is the first comprehensive treatment of spatial and temporal data warehouses. It is a starting point for newcomers, and an orientation point for professionals in the coexisting jungle of incomplete research results and real-life mandated, but not well established, applications.[2]

La Dra. Malinowski había planteado la posibilidad de contar con nuevos materiales de estudio en español sobre almacenes de datos pues había constatado que los libros de texto utilizados por dicha Universidad que provienen de España y México ocasionaban confusión debido a la falta de uniformidad en la terminología que los estudiantes de la UCR estaban empleando en el aula. Además, en algunas ocasiones, los términos mexicanos eran más utilizados que los españoles y en otras eran los españoles, dependiendo del término. En otras palabras, la Universidad necesitaba contar con materiales en español con terminología uniformada que se ajustara al uso real dentro del aula. La elaboración de un glosario con definiciones y la traducción del capítulo 3 del libro publicado por Malinowski y Zimányi se seleccionaron para la elaboración de esta tesina a solicitud de la Dra. Malinowski, pues responde a esta necesidad de uniformar terminología,

[2] William Harvey Inmon es considerado universalmente como el padre de los almacenes de datos (véase http://www.inmoncif.com/about).

especialmente para las clases de posgrado que ella imparte en la UCR. El capítulo 3 se escogió debido a que incluye los conceptos fundamentales de los almacenes de datos.

1.2 Objetivos

El objetivo de esta tesina es doble: traducir el capítulo 3 del libro en cuestión con el fin de generar un producto para uso de la UCR e incursionar en el campo de la traducción técnica relacionada con almacenes de datos. Para el primer componente, la meta es generar un producto final que se convierta en un instrumento educativo para los cursos de posgrado de la UCR; es decir, va más allá de ser simplemente una tesina pues satisfará una necesidad sentida e importante en dicha institución. Con respecto al segundo componente, este proyecto me permite realizar el trabajo final sobre un tipo de traducción de mucho interés para mí, puesto que al tratarse de una traducción técnica tengo la posibilidad de adquirir nuevos conocimientos y aplicar técnicas que me servirán para desempeñar mejor mi actual trabajo como Asistente de Traducción del Banco Interamericano de Desarrollo (BID) y, como consecuencia, mis superiores pueden confiar en que puedo llevar a cabo un trabajo de índole técnica con la seguridad y conocimientos suficientes para entregar un producto de alta calidad. En el BID lidiamos con todo tipo de traducciones técnicas, no solamente económicas y financieras, y la práctica derivada del presente trabajo y los conocimientos adquiridos pueden ejercer una influencia positiva en mi labor diaria como traductora e investigadora de términos.

1.3 Metodología

El primer paso consistió en traducir el glosario provisto por la Dra. Malinowski, el cual forma parte de su libro. Luego de varios meses de investigación, traducción y revisión se llegó a un producto final. El glosario sirvió de base para iniciar el capítulo 3, que incluye muchos de los términos ahí descritos.

1.3.1 Búsqueda de información

Para cumplir con el objetivo fue necesario llevar a cabo una amplia labor de investigación basada fundamentalmente en consultas avanzadas en la Internet para identificar términos y conceptos clave con miras a lograr una buena comprensión del texto a traducir. La búsqueda de términos

fue una tarea continua incluso después de elaborado el glosario, puesto que, como ya se mencionó, en el texto surgieron otros términos que no se habían contemplado anteriormente. Dicha labor requirió un mínimo de cuatro a cinco búsquedas por término para confirmar el uso más común de la palabra según diferentes fuentes. De hecho, muchas de las palabras tenían traducciones diferentes dependiendo del país. Un claro ejemplo es el término "almacén de datos" que se explica más adelante (véase la sección 5.1.1). Sus posibles traducciones eran "almacén", "bodega" o "depósito". Para este caso en particular se realizaron alrededor de 15 búsquedas de Internet para poder decidir cuál era el término más apropiado para nuestro contexto, considerando siempre al estudiante y profesor costarricense como nuestra "audiencia" meta. Vale decir que este término y varios otros requirieron además una búsqueda en diccionarios tanto bilingües como monolingües y de sinónimos con el fin de confirmar su significado o proponer más de una opción a la Dra. Malinowski.

1.3.1.1 Elaboración del glosario

La traducción del glosario (con definiciones) contenido en el libro de la autora fue el primer paso para realizar esta traducción que, por su naturaleza técnica, recoge un sinnúmero de términos no sólo desconocidos por la estudiante sino también dispersos en diferentes fuentes y, por lo tanto, heterogéneos. El glosario fue preparado en un lapso de tres meses con la idea de que me sirviera para darme una idea de lo que implicaría traducir un texto técnico de este tipo, a la vez que serviría también para las clases de posgrado de la UCR una vez revisado y aprobado por la autora, con miras a uniformar la terminología usada por ellos.

A pesar de haber producido una "versión final" del glosario antes de iniciar la traducción, me di cuenta de que en el capítulo 3 había algunos términos no incluidos en el glosario y sugerí a la Dra. Malinowski revisarlos e indicarme si le parecía conveniente incluirlos. Para ello, recurrí nuevamente a la Internet con el fin de buscar también definiciones, lo cual le ayudaría a la autora a decidir y darse cuenta de lo que está publicado en Internet. Los términos nuevos que se incluyen como Anexo 3:

1.	Conteo doble
2.	Dimensión de valores múltiples
3.	Dimensión degenerada
4.	Dimensión multivaluada

5.	Dimensión que juega diferentes roles
6.	Jerarquía alternativa
7.	Jerarquía balanceada
8.	Jerarquía desigual
9.	Jerarquía disjunta
10.	Jerarquía estricta
11.	Jerarquía no balanceada
12.	Jerarquía no cubierta
13.	Jerarquía no estricta
14.	Jerarquía non-onto
15.	Jerarquías definidas por el usuario o jerarquías de niveles múltiples
16.	Jerarquías paralelas dependientes
17.	Jerarquías paralelas no dependientes
18.	Jerarquías que omiten niveles
19.	Jerarquía recursiva
20.	Mapeo de datos
21.	Miembro no hoja
22.	Operación de mezcla
23.	Tabla plana
24.	Tabla tipada

Estos términos se incluirían en una nueva versión del capítulo III del libro, pues la autora consideró que requería más tiempo para revisarlos y, posiblemente, aprovechar para incluir otros.

1.3.2 Traducción del texto

La traducción técnica se llevó a cabo sobre la base del conocimiento adquirido en la traducción del glosario y la investigación de los términos en la Internet, los cuales fueron además corroborados o editados por la Dra. Malinowski. El trabajo de traducción se dividió en tres partes: traducción de las secciones 3.1 a 3.2., de 3.3 a 3.5 y de 3.6 a 3.9. Esto permitió establecer un orden en la revisión del texto por parte de la autora (y facilitar el trabajo de ambas partes) de manera que una vez que completé y revisé la primera parte, envié el texto a la autora para revisión. Una vez revisado y aprobado el texto de esta primera parte se inició la traducción de la segunda parte, con el fin de tomar en cuenta la edición y los comentarios de la autora, lo cual facilitó en gran medida mi labor de traducción y la revisión de ella para las secciones subsiguientes.

Quizás valga la pena resaltar aquí las dificultades de realizar una traducción técnica, aún cuando se cuenta con la asesoría directa de un experto. En las secciones en las que se presentaban los

códigos de Visual Basic, inicialmente en un intercambio con la autora se había entendido que los códigos de programación no requerían traducción y que si se modificaba alguno de ellos el programa no se ejecutaría. No obstante, durante la segunda revisión del documento la traductora indicó que los códigos sí debían incluir algunas palabras en español los cuales además deberían coincidir textualmente con las gráficas que se referían a ellos. En el caso de los códigos de programación, quizás de manera caprichosa para los no expertos, en algunas instancias el texto requería traducción y en otras no; desafortunadamente eso ya es un problema de la programación en sí que sólo los programadores conocen bien. En el caso de los gráficos, era importante mantener la congruencia de terminología entre texto y gráfico para facilitar la lectura al estudiante o profesor —y obviamente mantener la coherencia en todo el documento. Fue interesante además poder tener la oportunidad de llevar también un curso de Visual Basic durante la preparación de este proyecto que me ayudó a entender un poco la lógica de lenguaje de programación y del significado de algunos códigos. A continuación se presenta un ejemplo del código original en inglés y los ajustes correspondientes en español:

alter table Customer add constraint CustomerTypeCK check (CustomerType in ('Person', 'Company'))	alter table Cliente add constraint VerTipoCliente check (TipoCliente in ('Persona', 'Compañía'))

Por último, dentro del texto (párrafos) hubo la necesidad de insertar algunos símbolos especiales que a veces no facilitaban mucho el ajuste del formato; por ejemplo, ⊜ y su ubicación dentro del texto lo determina Word de forma caprichosa.

1.3.3 Traducción de figuras

Tras finalizar la traducción y revisión de todas las partes, las uní en un solo documento e inicié la traducción de las figuras. Para ello fue necesario aprender a utilizar Microsoft Visio —programa informático de dibujo que permite, entre otros, dibujar diagramas de bases de datos. Esta parte del proyecto fue relativamente sencilla pues al haber terminado la traducción en su totalidad ya los términos habían sido asimilados y luego sólo fue cuestión de verificar que se utilizaran de manera uniforme en el texto y las figuras. No obstante, lo que sí requirió más tiempo fue el ajuste de las figuras al texto en español que, obviamente, era más largo que el inglés en la mayoría de

las ocasiones y también en algunos casos se debió reducir la fuente de la letra a un tamaño menor pero legible para poder incorporarlo sin afectar la forma de las figuras. Asimismo, fue necesario manipular los elementos gráficos de las figuras para poder hacer calzar la nueva información en español.

Por último, una vez insertadas las figuras en el texto revisado llevé a cabo una revisión final contra el original y se lo envié a la autora para revisión y aprobación final.

2 Almacenes de datos

2.1 Historia de los almacenes de datos

En esta sección se hace un breve repaso a la evolución de los almacenes de datos, comenzando por sus primeros pasos en los años sesenta, momento en el cual el procesamiento de la información era bastante complejo, hasta llegar a lo que hoy día conocemos como almacenes de datos: un sistema automatizado que permite obtener información confiable y oportuna gracias al avance que se ha hecho durante los últimos años en el desarrollo de programas informáticos.

Wesley Romão et al. (1995) hacen mención al hecho de que el uso de las computadoras ha traspasado la frontera del ámbito técnico para penetrar en los más variados ámbitos de la actividad comercial. Las computadoras y los sistemas de comunicación no sólo son responsables del cambio en la naturaleza del trabajo de las personas sino que también han reformulado el mundo de los negocios. El concepto tradicional de trabajo que involucra grandes cantidades de papel, informes complicados y toma de decisiones basadas en hechos poco concretos es hoy por hoy un concepto de los sistemas de la información mucho más moderno. Este concepto reúne el uso de múltiples tecnologías con el fin de informar de manera coherente y directa sobre lo que es verdaderamente relevante.

Las aplicaciones informáticas se modificaron a partir del procesamiento de transacciones y actividades de seguimiento que dominaron la industria en los años sesenta y setenta, en su etapa de análisis del problema, y en la década de los noventa, en su etapa de implementación de las soluciones encontradas. Los avances tecnológicos en equipos y programas de computación, así como en las telecomunicaciones, responden a varias de estas interrogantes mediante el uso de

nuevas aplicaciones en el trabajo del personal ejecutivo de una empresa. Las tecnologías computacionales para el análisis de información gerencial son un elemento fundamental para el éxito de las empresas. Durante muchos años los administradores consideraban que la toma de decisiones era un arte, es decir, un talento que se adquiría después de un largo tiempo gracias a la experiencia (aprendizaje por ensayo y error) y su labor gerencial era también considerada un arte porque era posible emplear una gran variedad de estilos individuales para resolver problemas de la más variada naturaleza en múltiples áreas de negocio. Con frecuencia, estos estilos se basaban en la creatividad, el buen juicio, la intuición y la experiencia, y no en métodos estructurados. No obstante, el entorno en el cual se llevaban a cabo estas labores se fue modificando de manera acelerada. Los negocios y su ambiente son hoy en día más complejos.

De acuerdo con Turban (1994) cada vez es más difícil tomar decisiones por dos motivos: primero, el número de alternativas disponibles es mucho mayor que antes en virtud de la aparición de mejores tecnologías y sistemas de comunicación. Segundo, el costo de los errores puede ser muy elevado debido a la complejidad y la magnitud de las operaciones de automatización y de la cadena de reacción que un error puede ocasionar en muchas partes de la organización. Justamente por eso, los beneficios pueden ser muchos si se adoptan las decisiones acertadas.

Además de la complejidad del ambiente, la sociedad está viviendo tiempos de competitividad global. Los sistemas de información son herramientas importantes en esta tendencia mundial, ya que pueden simplificar la individualización de productos y personalizar aún más los servicios. Fue dentro de este contexto que el término *data warehouse* fue acuñado a finales de la década de los setenta como una solución al problema de la falta de información adecuada para el apoyo a la toma de decisiones por parte de los diferentes niveles jerárquicos de una organización.

2.2 Definición

Encontrar una definición de almacenes de datos que pueda ser entendida por cualquier persona es un poco difícil si nos referimos a libros especializados o incluso a la Internet. Si bien la Wikipedia no es considerada una fuente plenamente confiable para un trabajo académico de este tipo, me pareció importante demostrar que en la Internet la definición que se encuentra en casi

todos los sitios que hablan del tema la definen, casi que sin excepción, de la manera en que lo hace la Wikipedia (2010). Esta definición está basada en la descripción ofrecida por William Inmon:

> En el contexto de la informática, un almacén de datos (del inglés *data warehouse*) es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

Se requiere más de una búsqueda para lograr entender el concepto. Al igual que la Wikipedia, muchos sitios virtuales académicos ofrecen una versión similar a la anterior. Por suerte, tras la lectura de varios documentos en línea, sí fue posible encontrar algunos que ofrecen definiciones más claras y entendibles para los que no estamos familiarizados con el tema, tal y como lo hace Roberto Velasco (s. f.) al describirlo en los siguientes términos:

> Un almacén de datos (o *data warehouse*) es una gran colección de datos que recoge información de múltiples sistemas […] y cuya actividad se centra en la toma de decisiones —es decir, en el análisis de la información, en vez de su captura. Una vez reunidos los datos de los sistemas se guardan durante mucho tiempo, lo que permite el acceso a datos históricos; así, los almacenes de datos proporcionan al usuario una interfaz consolidada única para los datos, lo que hace más fácil escribir las consultas para la toma de decisiones.

La Dra. Malinowski (comunicación personal, 2010) lo define de manera aún más ilustrativa:

> Para profundizar en su definición es necesario entender primero que las bases de datos son una colección de datos relacionados lógicamente donde se especifican los objetos y las relaciones que existen entre ellos. Por ejemplo, podemos definir un objeto "Persona" y otro "Automóvil" y establecer una relación lógica entre ellos para indicar cuál persona es dueño de cuál auto. Los almacenes de datos usados en los sistemas de apoyo al proceso de toma de decisiones son bases de datos con una estructura especial definida por el modelo multidimensional. Esta estructura facilita el almacenamiento de datos históricos almacenados durante largos períodos de tiempo en contraste con las bases de datos tradicionales —llamados también operacionales o transaccionales— que se usan, por ejemplo, en los bancos. Además, esta estructura facilita la expresión y ejecución de consultas muy complejas que usan datos agregados o resumidos, por ejemplo, ventas anuales en lugar de ventas específicas, o ventas por provincia en lugar de ventas particulares en cada tienda. En síntesis, los almacenes de datos son un repositorio de datos relacionados entre sí.

Curiosamente, la Wikipedia en inglés (2010), a diferencia de la de español, ofrece una definición más clara y concisa, también ofrecida por Inmon: “A data warehouse is a repository of an organization’s electronically stored data designed to facilitate reporting and analysis”. Claro está que hay mucho más detrás de estas definiciones, pero para los propósitos de este trabajo considero que las aquí presentadas explican el concepto en palabras que cualquier persona no experta pueda entender.

3 Traducción técnica

3.1 Importancia y definición

Al explicar la importancia de la traducción técnica, Byrne (2006:3-6) explica que un 90% de la producción anual total de traducciones en el mundo corresponde a esta esfera de trabajo. A pesar de este alto nivel de demanda existen ciertos mitos acerca de este tipo de traducción:

- *La traducción técnica abarca temas económicos, legales, comerciales, etc.:* La palabra "técnica" se refiere a la tecnología y textos tecnológicos pero el hecho de que se trate de una terminología especializada no la hace necesariamente técnica. Por ejemplo, la religión tiene terminología específica pero no es técnica. En esencia, la traducción técnica lidia con textos sobre temas basados en el conocimiento aplicado (como el tema que nos concierne).
- *La terminología es lo más importante:* Si bien la terminología es el aspecto más notable de un texto técnico y es un medio para transmitir información, la terminología representa sólo entre un 5% y 10% del contenido total de textos técnicos (Newmark 1998).
- *El estilo no es relevante:* Byrne explica que esto raya en lo ofensivo, pues implica que los traductores técnicos no poseen las mismas destrezas lingüísticas y de redacción que otros traductores. Agrega que si vemos el estilo como la forma en que escribimos, las palabras que usamos y la forma que construimos las oraciones, entonces el estilo es igualmente, si no más importante en la traducción técnica que en otras áreas, porque está allí por una razón y no simplemente por razones artísticas o de entretenimiento.
- *La traducción técnica no es creativa; es sólo un proceso de transferencia:* Para poder transmitir información de manera adecuada y eficaz, los traductores técnicos deben encontrar soluciones innovadoras y creativas y así garantizar una comunicación satisfactoria.
- *Es necesario ser experto en un campo altamente especializado:* Se dice que uno no podría especializarse en más de una o dos áreas, pero Byrne subraya que si el traductor cuenta con un sólido conocimiento de los principios básicos y las tecnologías bien podría "disimular" que lo es, es decir, el traductor debe poseer conocimientos acerca del tema en cuestión para saber cómo lidiar con un texto o para poder adquirir cualquier otra

información que necesite. Cuando se investiga un tema nuevo siempre es más fácil tener una idea básica de lo que se trata a no tener ningún conocimiento. Por ello es fundamental que el traductor técnico posea excelentes destrezas de investigación.

- *La traducción técnica sólo transmite información especializada:* Si bien la preocupación principal es transmitir información de manera precisa también es fundamental que la información (i) se presente en forma correcta, (ii) esté completa y (iii) pueda utilizarse de manera apropiada y eficaz.

Es interesante mencionar aquí el caso del Sindicato Nacional de Traductores (SINTRA) de Brasil, que en años recientes decidió retirar el término "técnica" de la modalidad antes llamada por ellos como 'tradução técnica' y conocida ahora solamente como 'tradução', con el fin de dar a los traductores las mismas oportunidades de retribución ante las editoras y dar un mismo valor al trabajo traductor, sin importar su nombre. Así lo explica Eloisa Barbosa, representante de dicho Sindicato, (2005:5):

> Com essa alteração, espera-se demonstrar que toda tradução é «técnica» (mesmo uma carta de amor pode ser muito técnica) e proporcionar um maior poder de negociação ao tradutor que trabalha para editoras.

A pesar de este enfoque innovador por parte de los brasileños, la traducción técnica sigue teniendo una diferenciación con respecto a otros tipos de traducción cuando se trata de describirla y enmarcarla dentro de un concepto concreto. Sándar Hervey et al. (1995:154) ofrecen una definición sencilla y al punto, a la vez que agregan cierta validación al criterio de los brasileños:

> By 'technical' translation we mean especially the translation of empirical/descriptive texts written in the context of scientific or technological disciplines". […] It is also worth emphasizing that the problems met in translating technical texts are to a great extent no different from those met in translating in any genre, specialized or not.

Ahora bien, la traducción técnica no sólo ha logrado alcanzar en gran medida a la literaria en importancia, sino que pareciera equiparse ya con ella en valor, diferenciándose únicamente en su mayor pragmatismo (Guidère, 2008:36):

> la traduction gagne de plus en plus de terrain face à la traduction littéraire, et les dictionnaires spécialisés se multiplient et se diversifient. […] La traduction occupe définitivement une place importante dans la société moderne. Contrairement à la traduction littéraire […] elle remplit une fonction économique et sociale indispensable…

Por último, Byrne (2006:23-24) la define en palabras sencillas como un proceso comunicativo, un servicio suministrado a nombre de otra persona para cumplir con un propósito específico dentro de una situación y entorno particular. En pocas palabras, la traducción técnica es un servicio comunicativo orientado al lector. El objetivo de la traducción técnica, según Byrne, es presentar información técnica a una audiencia nueva y no simplemente reproducir el texto fuente *per se* o reflejar su estilo o lenguaje. Se trata de brindar un servicio comunicativo en respuesta a una demanda específica de información que el cliente pueda acceder fácilmente en términos de coherencia, claridad y rapidez de entrega del producto.

3.2 El papel del traductor

Con el propósito de recopilar datos acerca de la demanda de la traducción técnica en el mercado brasileño, Polchlopek y de Abreu Alo de la Universidade Federal de Santa Catarina (UFSC/PGET) (2009:108), realizaron una encuesta a clientes y traductores. Algunas de las personas entrevistadas eran de la opinión que el traductor es una persona capaz de transmitir en una lengua lo que entiende en otra, con cohesión y sentido, sin alterar el original. Otras lo describieron como el profesional habilitado para traducir. Las autoras explicaron en sus conclusiones que a pesar de su importancia, especialmente en lo que se refiere a los textos técnicos, la responsabilidad principal del traductor se vincula con la definición de estrategias, decisiones y gestión que van desde su competencia lingüística y cultural hasta la evaluación de la función (*skopos*) que el texto traducido pretende alcanzar en la cultura meta. Más allá del simple dominio de códigos y contextos de producción y recepción, el traductor debe tener una "percepción aguda, creatividad, sensibilidad y experiencia de traducción" (ibid:109). En este sentido, las autoras defienden la idea de que la traducción sea hecha por traductores profesionales, aunque no sean especialistas en sus áreas de trabajo y que el técnico (ingeniero, médico, economista) asuma la tarea de revisión terminológica y las cuestiones de orden estilístico.

Por su parte, Byrne (2006:15) equipara el papel del traductor técnico con el del escritor técnico, pues ambos usan información procedente de diversas fuentes, no sólo del texto original, para producir un texto meta que sea eficaz y que realice la función comunicativa deseada. Agrega que lo que distingue a un buen traductor técnico es la habilidad de hacer algunas de las cosas que un

escritor técnico hace con el fin de asegurarse de que la persona que en última instancia lee el texto pueda hacerlo con relativa facilidad y que cualquiera que sea la tarea que el lector debe llevar a cabo ésta le resulte practicable tras la lectura del texto.

3.3 Teoría de la traducción técnica

Byrne (2006:22-23) explica que cuando se trata de explicar o ubicar a la traducción técnica dentro de un marco teórico a menudo es extremadamente difícil saber por dónde empezar. Esto se hace mucho más problemático si se toman en consideración el elevado número de enfoques, modelos, reglas y teorías que existen acerca del tema, que muchas veces se contradicen entre ellos. Byrne también se lamenta del hecho de que la traducción técnica ha sido dejada de lado en las deliberaciones teóricas y que pocas veces se la menciona de forma específica.

Sobre la base de su definición de la traducción técnica como "servicio comunicativo orientado al lector", Byrne se apoya en la teoría de Skopos según la cual un proyecto de traducción particular podría requerir una traducción libre o fiel o un rango intermedio entre libre y fiel, dependiendo el propósito de la traducción. Dentro de un mismo texto podría ser necesario cambiar de un enfoque literal a uno libre con respecto a diferentes frases, oraciones o párrafos. Por lo tanto, la mejor manera de enmarcar la traducción técnica, según Byrne, es adoptar un enfoque comunicativo tal como lo propone la teoría de Skopos, pues es el enfoque que verdaderamente acoge la realidad profesional de la traducción y las necesidades, expectativas y obligaciones del traductor técnico.

Vale aclarar aquí que el presente proyecto, en mi opinión, se ajusta muy bien a lo indicado por Byrne, pues el tema de almacenes de datos cumple con dos aspectos que él menciona: Primero, esta traducción, por su naturaleza especializada, estaría en un punto intermedio entre literal —por requerir el uso de formas preestablecidas por la terminología— y libre —en lo que se refiere a la parte estilística del texto en la que puede haber mayor libertad de redacción (véase la sección 5.1). Segundo, la función o propósito de esta traducción es comunicar cierta información de manera que el "usuario" o lector pueda poner en práctica fácilmente lo que ahí se explica (en nuestro caso, por cualquier estudiante o profesor de la UCR).

3.4 Teoría de Skopos

La teoría de Skopos (propósito) es un enfoque funcionalista a la traducción desarrollado en Alemania a finales del decenio de los setenta Hans J. Vermeer, con la contribución de Katharina Reiss (Baker 2001:235). De acuerdo con Vermeer (1989:20), la traducción se realiza para diferentes clientes con un propósito o propósitos específicos en una situación determinada y el traductor debe cumplir el trabajo encomendado teniendo en mente dicho propósito. La solicitud del cliente es un prerrequisito fundamental para que se cumpla el objetivo de Skopos y, según Vermeer, la meta de todo proceso traductor y las condiciones bajo las cuales se alcanza dicha meta deben ser negociadas entre el cliente y el traductor. El texto meta es "funcional" y colma las expectativas y necesidades de la audiencia meta. En palabras de Vermeer, la regla de Skopos consiste en:

> "[T]ranslate/interpret/speak/write in a way that enables your text/translation to function in the situation in which it is used and with the people who want to use it and precisely in the way they want it to function."

Christiane Nord (1997:29) resume la regla de Skopos con la expresión "El fin justifica los medios". Asimismo, existen dos reglas generales de esta teoría: la coherencia y la fidelidad. La regla de la coherencia estipula que el texto meta debe conformarse a la cultura lingüística y situación comunicativa en las cuales se va a utilizar dicho texto meta. La regla de la fidelidad determina, por su parte, que debe existir una congruencia intertextual entre el texto fuente y el texto meta. No obstante, el grado y forma de la fidelidad depende del propósito del texto meta y la comprensión del traductor del texto fuente. Sin insistir mucho en alcanzar la meta de la traducción perfecta o en una estrategia determinada, los funcionalistas abogan por que los traductores traten de encontrar soluciones óptimas de conformidad con la situación que se les presente. Pueden elegir ser fieles al texto del original y preferir una estrategia literal, o bien agregar, borrar o modificar información según lo consideren apropiado, dependiendo de las condiciones culturales y las necesidades de la audiencia/cliente (Gentzler, 2001:70-71).

De esta manera, el traductor despoja al texto meta de su estatus básico y sagrado (Aveling 2002) si considera que ello es necesario con el fin de alcanzar el Skopos trazado. El texto meta simplemente es una de varias fuentes de información utilizadas por el traductor y no constituye

el criterio principal en el proceso de decisión de un traductor. Tal como lo menciona Hönig (1998:9)

> "…the source text should no longer be seen as the 'sacred original,' and the purpose of the translation can no longer be deduced from the source text, but depends on the expectations and needs of the target readers."

Por último, en la teoría Skopos la forma en que se pretende que un texto meta sea recibido básicamente determina cuál estrategia de traducción es la más apropiada. Existen tres tipos primordiales de propósitos: (a) el propósito general de por qué el traductor lleva a cabo la traducción; (b) el propósito comunicativo (por ejemplo, informar); y (c) el propósito estratégico del procedimiento que se ha elegido utilizar (por ejemplo, traducción literal o libre). (Hatim 2001).

4 Terminología

4.1 Definición y diferenciación de términos

Para este análisis es necesario definir algunos términos clave del campo de la terminología con el fin de enmarcarlo desde el punto de vista de la elaboración del glosario y la terminología empleada en la preparación de la traducción. No obstante, es importante aclarar que las siguientes delimitaciones no pretenden abarcar todo el cosmos de términos relativos a la terminología ni entrar en detalle en ellos, sino más bien presentar la diferenciación y relación entre algunos conceptos de uso común en el estudio de la terminología. De hecho, también es pertinente resaltar aquí que parece ser evidente que cada autor hace uso de las palabras según sus propósitos y algunas de ellas, por lo tanto, podrían dan lugar a debates interminables que no tendrían cabida aquí.

Para poder definir la terminología es importante notar que varios expertos han tratado de definirla y diferenciarla con respecto a otros conceptos tales como lexicografía y lexicología. De acuerdo con el *Diccionario de la Real Academia*, se definen así:

a. Lexicografía: (De lexicógrafo). 1. f. Técnica de componer léxicos o diccionarios.
b. Lexicología: (Del gr. λεξικόν, léxico, y -logía). 1. f. Estudio de las unidades léxicas de una lengua y de las relaciones sistemáticas que se establecen entre ellas.
c. Terminología: 1. f. Conjunto de términos o vocablos propios de determinada profesión, ciencia o materia.

De lo anterior se desprende que la lexicografía se refiere a la creación de términos, la lexicología al estudio de los mismos y la terminología a la agrupación de todos ellos. Igualmente, como se mencionó anteriormente, la terminología lleva inherente la idea de que no se trata de palabras comunes y corrientes, como sí lo implican los dos primeros, sino más bien de palabras específicamente utilizadas en una disciplina determinada. De acuerdo con Sager (1990:2), la terminología y la lexicografía comparten objetivos comunes pero esta última además recopila datos sobre el léxico de un idioma a la vez que brinda información y, en algunos casos, un servicio o asesoría a los usuarios de un idioma.

Por su parte, Kyo Kageura (2002:9) en su libro *The Dynamics of Terminology* también resalta la importancia de hacer la diferencia entre terminología y término al abordar un análisis de este tipo, no sin aclarar que la búsqueda de definiciones en este campo puede ser controvertido y provisional si se toma fuera de contexto. Al respecto, se remite a las definiciones aportadas por Bessé, Nkwenti-Azek & Sager (1997) según las cuales

- término es la unidad léxica que consiste en una o más palabras que representan un concepto dentro de un dominio y
- terminología es el vocabulario de un campo de estudio.

Kageura hace mención a la importancia de diferenciar entre término y palabra, ya que algunos autores usan esta última como sinónimo de unidad léxica, lo cual se presta a confusión puesto que el propósito de un glosario es distinguir entre las palabras y los términos. Del mismo modo, cuando se hace referencia a un término estamos hablando de una unidad léxica mientras que cuando se hace referencia a una terminología estamos hablando de una colectividad de términos —en comparación con unidad léxica (palabra) y vocabulario.

Asimismo, según Cabré (1992:169) el término y la palabra se distinguen por cuestiones pragmáticas o comunicativas:

> Aparentemente, los términos no parecen diferir mucho de las palabras, considerados desde una perspectiva formal o semántica; pero se diferencian notablemente de ellas si aportamos criterios pragmáticos y comunicativos. En efecto, la peculiaridad más notable de la terminología, en contraste con el léxico común, consiste en que sirve para designar los conceptos propios de las disciplinas y actividades de especialidad.

Una buena definición que abarca varios conceptos importantes de lo que es realmente la terminología la ofrece Sager (1990:2):

> Terminology is the study of and the field of activity concerned with the collection, description, processing and presentation of terms, i.e. lexical items belonging to specialized areas of usage of one or more languages.

Por otro lado, Alain Rey (2009:135) adopta la palabra *terminografía* como término para definir la terminología práctica y la diferencia de ésta en que su función es compilar, describir y controlar un conjunto de términos mediante métodos y procedimientos que son en parte originales y en parte prestados de la lexicografía, la documentación y la traducción. Su objetivo

primordial es describir un objeto que puede ser una terminología (es decir, un conjunto de términos) una nomenclatura o una estructura compleja de sistemas y subsistemas de términos y conceptos en un campo de estudio. Asimismo, Rey comenta sobre la dificultad en encontrar una definición: "Regarding 'terminology', the history of this word is even more recent, its definition even less precise, and its uses so diverse that one would wish for some regularization." (2009:126).

4.2 Terminología y su normalización (ISO)

La normalización terminológica, que se encuadra dentro del marco general de la normalización de productos, procesos, unidades de medida, etc., surge de la preocupación en el seno de la ciencia y la tecnología por las posibles confusiones en la utilización de los términos y el deseo de establecer criterios lingüísticos claros para la designación y la descripción de los conceptos. Ese afán de normalización se ve plasmado en el interés que ha habido por enmarcar a la terminología dentro de estándares internacionales. Así, por ejemplo, según la norma ISO 1087 (1990) un término es: "a conventional symbol for a concept which consists of articulated sounds or of their written representation. A term may be a word or a phrase".

A continuación se presenta una lista de algunas normas ISO relacionadas con la terminología:

- ISO 639: 1988 – Código para la representación de nombres de idiomas
- ISO 704: 1987 – Principios y métodos de la terminología
- ISO DIS 860: 1993 – Armonización internacional de conceptos y términos
- ISO 1087: 1990 – Vocabulario de terminología
- ISO 2145: 1978 – Documentación: Numeración de divisiones y subdivisiones en documentos escritos
- ISO 10241: 1992 – Normas terminológicas – Preparación y presentación

La motivación fundamental en la creación de estas normas ha sido ayudar a organizaciones y a empresas a crear, homogeneizar y gestionar sus terminologías, así como a mejorar la información contenida en los estándares técnicos propuestos. De esta forma se facilita no sólo la comunicación entre especialistas, sino que es el mejor camino para conseguir que los estándares propuestos en otros ámbitos científicos y tecnológicos se apliquen. Por otra parte, la creación de

estándares asegura la armonización no sólo en el contenido de los repositorios terminológicos sino también en su forma, es decir, la manera en la que la información está codificada, ya que un formato estándar asegura la posibilidad de evaluar y validar la información contenida en los repositorios terminológicos, el intercambio de recursos terminológicos y la reutilización de los mismos, aspectos que hacen de la terminología un elemento de importancia central para el desarrollo y buen funcionamiento de la sociedad actual (Pérez, 2002).

Un tema que ha surgido en años recientes es la planificación terminológica. Se han configurado diferentes comisiones lingüísticas que han intentado orientar la formación de términos y existe la tendencia de rechazar la noción de que la terminología técnica y científica debe ajustarse al idioma inglés. Sager (1990:88-89), por ejemplo, brinda una lista de algunos de los requisitos que se consideran ideales en la traducción y que se han originado como resultado de esa preocupación internacional de controlar la proliferación de la terminología y establecer reglas o lineamientos para dar nombre a objetos y conceptos. Dicha lista explica que todo término:

a. debe tener relación directa con el concepto. Debe expresar el concepto claramente. Es preferible que tenga una construcción lógica;
b. debe ser sistemático desde el punto de vista léxico. Debe seguir un patrón léxico existente y, si las palabras son de origen extranjero, debe preservarse una transcripción uniforme;
c. debe conformarse a las reglas generales de formación de palabras del idioma que también dictará el orden de las palabras en las estructuras compuestas y frases;
d. ha de brindar formas derivadas;
e. no ha de ser pleonástico (es decir, no redundante, por ejemplo, combinar una palabra extranjera con una palabra nativa que tenga el mismo significado);
f. Sin sacrificar precisión, debe ser conciso y estar desprovisto de información innecesaria;
g. no ha de ser sinónimo absoluto, relativo ni aparente;
h. no ha de tener variantes morfológicas;
i. no ha de tener homónimos;
j. han de ser monosémicos;
k. el contenido de los términos ha de ser preciso y su significado no ha de superponerse a otros términos;

l. el significado del término ha de ser independiente del contenido.

Con base en lo anterior, no es de sorprender que el campo de la terminología tenga mucho que ofrecer por su naturaleza práctica y de servicio social y científico. Gracias a esta estandarización la terminología está muy bien posicionada para desempeñar un papel cada vez más preponderante no sólo en el ámbito de la traducción sino en la sociedad en general.

4.3 Terminología y ciencia de la información

Es interesante notar que tanto el campo de la terminología como el de la ciencia de la información se benefician mutuamente de la interacción entre ambos ámbitos. Por una parte, el uso de computadoras ha ayudado a sistematizar los miles y miles de términos que han surgido en cada disciplina de estudio, lo cual ha hecho que la labor del terminólogo se facilite en gran medida. Por otra parte, el campo de la terminología y la forma como se ha venido formalizando su estudio mediante conceptualizaciones y las teorías que la enmarcan han contribuido a un sinnúmero de ámbitos como el de la ciencia de la información, que hoy por hoy pueden realizar el trabajo de elaboración de diccionarios y glosarios de una forma organizada y de acuerdo con normas internacionales de calidad.

Sager (1990:5-6) señala que la terminología exhibe un número de similitudes con la ciencia de la información pues ambas tienen una edad semejante y han pasado por un proceso paralelo en búsqueda de su identidad. Las dos son relativamente jóvenes y todavía siguen expandiendo sus fundamentos teóricos y el alcance de sus aplicaciones, notablemente dentro de una teoría común de comunicación. Ambas sirven el propósito de facilitar la comunicación en lenguajes especiales. Incluso el enfoque a la lengua tiene claras similitudes: los terminólogos y los especialistas en ciencias de la información someten el lenguaje a procesos de regularización, unificación y estandarización en aras de una comunicación más eficaz.

Los objetivos prácticos de la terminología, es decir, alcanzar una mayor unidad, congruencia y claridad de expresión en la comunicación simplifica en gran medida el trabajo de los expertos en ciencias de la información. Ambos campos están a su manera relacionados con la planificación del lenguaje; el trabajo de indizadores, analistas, revisores y traductores se facilita gracias a la labor de planificación de dialectos especiales que llevan a cabo los terminólogos y reduce la

carga de trabajo de los planificadores del lenguaje informático y la divergencia de lenguajes informáticos de los dialectos relativos a temas específicos (Sager:1990-7).

5 Proceso de traducción

5.1 Objetivo

Durante el proceso de elaboración de este trabajo se encontraron varias dificultades, muchas de ellas inherentes a la traducción técnica. En los próximos párrafos se presenta una clasificación de acuerdo con la naturaleza del problema, pero antes considero interesante plasmar aquí las opiniones de algunos traductores acerca de cómo inyectar mayor claridad a nuestras traducciones y mucho viene al caso de la traducción técnica, a la que a menudo se clasifica, de forma muy simplista y general, como un proceso fácil y "literal". Es precisamente este tema el que nos lleva a admitir que toda traducción puede ser mejorada con respecto al original y, volviendo al concepto teórico de Byrne, el objetivo de toda traducción técnica debe ser la comunicación satisfactoria reflejada en un producto que el lector pueda entender y utilizar fácilmente.

Laura Labella, en un aporte a la revista "Apuntes" (2002), escribió lo siguiente al defender la libertad de traducción en pos de perfeccionar la redacción, lo que algunos quizás llamarían "mejorar el original":

> Seguramente, los más ortodoxos pondrán el grito en el cielo y argüirán que se trata de una completa herejía. En mi opinión, lo importante es buscar la claridad y la eufonía. Si para hacerlo es necesario cambiar el orden de las oraciones de un párrafo, modificar la puntuación, ampliar una elipsis, eliminar o agregar palabras, hay que sentir la libertad de hacerlo siempre y cuando se respete el contenido del original. Ser fiel al original no quiere decir producir una traducción literal ni calcar la estructura gramatical del idioma fuente.

Por su parte, Maite Solana, ex Directora de la Casa del Traductor en Zaragoza, España, hace resonar esta idea con un punto muy válido que también se aplica a la traducción técnica:

> Me parece que a estas alturas los traductores estamos hartos de tener que repetir una y otra vez que una traducción literal de un texto no es una traducción. Es otra cosa. La traducción, para que pueda llamársela tal, no puede limitarse a la mera traslación mecánica, más o menos palabra por palabra o frase por frase, del original; la verdadera traducción requiere invertir la sintaxis, cambiar la puntuación, recrear imágenes, buscar expresiones que en la lengua de llegada signifiquen lo mismo que en la lengua original, aunque sea utilizando otras palabras, etc. Y trabajar de este modo no tiene nada que ver con ser infiel al texto. Precisamente la mejor manera de ser fiel a un original es no ser fiel a su literalidad en absoluto.

5.2 Glosario

La elaboración del glosario requirió muchas horas de trabajo de investigación. A continuación se muestran algunos ejemplos de la tarea de consulta de la Internet y la manera como la

Dra. Malinowski podía verificar mis sugerencias de traducción de términos. En la mayoría de los casos se ofreció más de un enlace electrónico donde el término en cuestión era utilizado. Vale recordar que excepto por los 24 términos nuevos incorporados al final del proceso, las definiciones fueron traducidas a partir del glosario facilitado por la autora.

Ejemplo 1: (con explicaciones)

Análisis de datos: data mining Libro de Rob y Coronel **Minería de datos:** Trabajo de Espinoza, Umadrid Artículo que me gustó por el título: Data mining: torturando a los datos hasta que confiesen http://www.uoc.edu/web/esp/art/uoc/molina1102/molina1102.html http://es.wikipedia.org/wiki/Miner%C3%ADa_de_datos Aquí se defiende el uso de minería (blog del año 2005): http://www.proz.com/kudoz/english_to_spanish/business_commerce_general/1168625-data_mining_solution_en_este_contexto_dir%C3%ADamos_investigador.html Por ejemplo: "Debo comentar también que llevamos aproximadamente 3 años traduciendo cosas con relativa frecuencia sobre la "minería de datos". Cuando empezamos hace tres años no era un término universalmente aceptado, pero ahora está totalmente consolidado como demuestran los URLs que añadí" **Data mining:** Tesis de Nader, y http://www.monografias.com/trabajos/datamining/datamining.shtml	Proceso mediante el cual se analizan grandes cantidades de datos con el objeto de identificar relaciones, tendencias, patrones y asociaciones impredecibles o desconocidas que podrían ser de valor para la organización.

Ejemplo 2: (con término no apropiado para Costa Rica)

Restauración de datos: (data refreshing) Glosario del Depto. Cómputo U en Argentina http://www-2.dc.uba.ar/materias/so/datos/glosario.pdf No es glosario de DW pero me pareció útil mencionártelo: Refreshing : refresco, restauración. Mecanismo que se utiliza para las memorias que tienden a perder su contenido con el transcurso del tiempo y que consiste en restaurar el contenido de las mismas cada cierto intervalo. Trujillo lo llama "refresco" en documento APUNTES... pero en Costa Rica no sonaría bien.	Proceso por el cual se propagan las actualizaciones de las fuentes de datos al almacén de datos, con una frecuencia específica, a fin de suministrar información actualizada para el proceso de toma de decisiones. La actualización se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETL).

Ejemplo 3: (con término no encontrado en fuente confiable en Internet; traducción sugerida)

Nivel de separación: Not found. En algunos casos lo encontré como partición y fragmentación pero no era nada confiable y puede confundirse con partioning.	En el modelo MultiDim, se refiere al nivel en el cual empiezan dos rutas de agregación alternativa.
Esquema en estrella:	Esquema relacional para representar datos multidimensionales conformados

Ejemplo 4: (con término que mereció búsqueda más exhaustiva)

Nivel de hoja: (leaf level) http://msdn.microsoft.com/es-es/library/ms165911(SQL.90).aspx *Una jerarquía de miembros de atributo que contienen los siguientes niveles: un nivel de hoja que contiene cada miembro de atributo diferente (a cada miembro del nivel de hoja también se le conoce como miembro hoja), niveles intermedios si la jerarquía de atributo es una jerarquía de elementos primarios y secundarios, y un nivel (All) opcional que contiene el valor agregado de los miembros hoja de la jerarquía de atributo. A los miembros del nivel (All) también se les conoce como miembros (All). De forma predeterminada, una jerarquía de atributo está definida por cada atributo de dimensión. De forma predeterminada, las jerarquías de atributo son visibles.* Pero en este mismo documento también usan nodo hoja y nodo raíz: *Jerarquía equilibrada: Jerarquía de dimensiones en la que todos los nodos hoja están a la misma distancia del nodo raíz.* **Nodo hoja: (leaf level)** De: Libro Diseño Concepto Carlos Batini http://books.google.com/books?id=-DP-0puz338C&printsec=frontcover&dq=Dise%C3%B1o+conceptual+de+bases+de+datos+By+Carlo+Batini&cd=1#v=onepage&q=ra%C3%ADz&f=false Aquí también: HP Capacity Advisor versión 4.0: Guía del usuario Glosario 1 http://docs.hp.com/es/5992-1928/go01.html *"Dispositivo que se encuentra al final de un árbol de E/S dado, es decir, es el último dispositivo."* HP Capacity Advisor versión 4.0: Guía del usuario Glosario 2 *http://docs.hp.com/es/T8670-90004/go01.html* *Objeto en el nivel inferior de una vista de árbol gráfica. Los nodos de hoja no tienen nodos secundarios.*	Nivel en una jerarquía que no está relacionado con un nivel hijo, es decir, contiene la información más detallada. Compárese con nivel raíz.

Vale mencionar que en el ejemplo 4, a pesar de la búsqueda exhaustiva, la autora decidió traducir el término como "nivel hoja", que es como lo utilizan en Costa Rica. Por lo tanto, este es un ejemplo claro del problema que tiene la UCR de encontrar en la Internet materiales con terminología que ellos usan en la práctica.

5.3 Recopilación de la terminología

Con base en lo anterior, podemos decir que el presente trabajo ha requerido el empleo de ciertas técnicas que se describen en los siguientes párrafos, pero primero empezaré por describir el proceso seguido para tomar una decisión acerca del propio término de "almacén" de datos.

5.3.1 Almacén, bodega o depósito

Al formular la propuesta de trabajo a la autora, esta reaccionó diciendo que podría empezar por ayudarle a justificar el uso del término "almacén de datos" por encima de bodega o depósito de datos pues era tema de debate entre los profesores en ese momento. Para ello fue necesario recurrir a búsquedas en la Internet y consultas al diccionario para aclarar conceptos. Lo encontrado se ilustra en el siguiente cuadro:

	Almacén	**Bodega**	**Depósito**
Definiciones del DRAE[1]	1. m. Edificio o local donde se depositan géneros de cualquier especie, generalmente mercancías. 2. m. Local donde los géneros en él existentes se venden, por lo común, al por mayor. 4. m. Am. Tienda donde se venden artículos domésticos de primera necesidad. 5. m. Ec. Local para abrir tienda de comercio. 6. m. Col. Y Ec. Tienda de comercio. 7. m. Hond. Tienda en donde se venden telas y, a veces, ropa. 8. m. desus. Conjunto de municiones y pertrechos de guerra. 9. m. pl. Establecimiento comercial donde se venden géneros al por menor.	1. f. Lugar donde se guarda y cría el vino. 2. f. Almacén de vinos. 3. f. Tienda de vinos. 4. f. Establecimiento, generalmente industrial, para la elaboración de vinos. 5. f. Cosecha o mucha abundancia de vino en algún lugar. La bodega de Arganda, de Valdepeñas. 6. f. despensa (lugar donde se guardan los comestibles). 7. f. troj (espacio para guardar cereales). 8. f. En los puertos de mar, pieza o piezas bajas que sirven de almacén a los comerciantes.	1. m. Acción y efecto de depositar. 2. m. Cosa depositada. 3. m. Lugar o recipiente donde se deposita.
Sinónimos de mundo.es	comercio, negocio, depósito, factoría, local, cobertizo, barracón, tienda, nave	bodegón, despensa, cueva, silo, subterráneo, sótano, cava, cripta	provisión, precipitado, entrega, tanque, almacén, resguardo, garantía, acopio, recipiente, poso, sedimento, arsenal, custodia, embalse, yacimiento, consignación, almacenamiento, acumulación
Resultados arrojados por la Internet	26.800	1.600.000	10.800.000
Resultados en sitios virtuales en España[2]	4.400	46.600	2.350.000
Resultados en sitios virtuales en México[2]	1.350	2.930.000	2.280.000
Resultados en sitios virtuales en Argentina[2]	494	9	252.000

[1] Se excluyen algunas definiciones irrelevantes al tema y para facilitar la lectura.

[2] España, México y Argentina se escogieron por ser países que representan áreas geográficas de la lengua, según mi criterio.

Con base en esta información, en principio podría haberse elegido "depósito" según el uso en sitios de la Internet y de acuerdo con el diccionario de sinónimos del mundo.es. No obstante, la autora al final descartó "depósito" porque no quería confusiones con "depósito bancario" que, de hecho, se utiliza en ciertas secciones de su libro. Asimismo, descartó "bodega" porque en Costa Rica una bodega es un lugar desordenado y sucio y, precisamente, el almacén de datos en nuestro contexto se refiere a una organización "ordenada" de datos.

Por otra parte, si deseamos utilizar el concepto en su forma verbal, lo más natural sería decir "almacenar datos" y no "embodegar" o "depositar" datos. Por lo tanto, el concepto aceptado por ella fue "almacén de datos".

5.3.2 Explicación de términos

Un aspecto que no es poco común en el campo de la traducción técnica es el hecho de no contar con términos exactos en la lengua meta. En el caso siguiente se ilustra esta dificultad claramente. El término "role-playing dimensions" no puede traducirse literalmente como "dimensión de juego de roles" pues, según lo verificado por la autora, no transmite la idea exacta y lo correcto debe ser "dimensión que juega diferentes roles"[3].

Vale la pena anotar aquí también que la expresión es específica al campo de almacenes de datos, al menos en lo que se refiere al uso práctico en la UCR. Es decir, al querer "mejorar" la frase (por decirlo de alguna manera) de forma tal que no suene muy literal o apegada al inglés —es decir, "rol" para traducir "role" y "juega" por "play"— podría utilizarse expresiones que, si bien parecerían ser estilísticamente más correctas, como "desempeñar, cumplir" un "papel" o una "función", no se ajustarían al uso común de la UCR. Por ello, inicialmente cuando se quiso darle una traducción más castiza la autora no dudó en corregir el "error" y aclarar la razón por la cual debía traducirse "rol" y "jugar". Este es un claro ejemplo de que muchas veces en la traducción técnica y en otras especialidades lo que rige es el uso del día a día y no lo que podría parecer más apropiado según nuestras normas y de lo que hemos analizado aquí sobre el hecho de que la traducción técnica no tiene que ser necesariamente literal.

5.3.3 Significados dobles

Un buen ejemplo es la palabra "branch" que en el texto fuente se usa de manera intercambiable para calificar a "rama" y "sucursal". En el primer caso, "rama" se refería al área de especialización de una persona y, en el segundo, a una sucursal bancaria. Fue interesante durante el proceso de traducción ver cómo a veces algunos detalles se pasan por alto y en la redacción nos damos cuenta de que hay algo que no hace sentido. La discrepancia se hizo evidente

---

[3] El glosario ofrece una definición de "rol" en este contexto.

especialmente durante la traducción de las figuras, ya que si se describía una lista de "cualidades" de una persona, la palabra "sucursal" no calzaba del todo. Fue ahí cuando me di cuenta de que, efectivamente, se estaban refiriendo a dos cosas diferentes. Es evidente que esta diferencia no se hizo patente en la redacción del texto original sino hasta que se realizó la traducción y se hizo mención de ello a la traductora quien pidió específicamente cambiarlo en el español para evitar confusiones.

Otra palabra que requirió aclaración de la autora fue "view" que en el glosario se había traducido como "vista" (véase el glosario para la definición). A lo largo de este trabajo de traducción, sin embargo, fue obvio que la palabra "view" tenía dos significados dependiendo del contexto: "vista" y "representación". A continuación se dan dos ejemplos de ambas instancias:

- Much research has been done in the field of relational databases on improving query optimization, indexing, join operations, and view materialization, considering the particularities of data warehouse applications.
- Commercial relational database systems include extensions to represent and manage a multidimensional view of data.

A simple vista, y si cumplimos al pie de la letra con el glosario que fue preparado al inicio del proceso, y para una persona que no conozca el tema a fondo, ambos "view" podrían traducirse como "vista" si no se conoce la diferencia en los contextos. La autora fue enfática al aclarar que ambos se referían a cuestiones diferentes y que la traducción para ambos debería quedar como sigue:

- Se han realizado muchos estudios en el campo de las bases de datos relacionales sobre cómo mejorar la optimización de las consultas, la indexación, las operaciones de mezcla y la materialización de las vistas, considerando las particularidades de las aplicaciones de los almacenes de datos.
- Los sistemas de bases de datos relacionales comerciales [14, 101, 216] incluyen extensiones para representar y gestionar la representación multidimensional de datos.

Ciertamente, la traducción técnica requiere la verificación periódica de términos por parte del autor del texto fuente, con el fin de garantizar que la información que va a ser utilizada por el cliente sea veraz. No obstante, es desafortunado que en la práctica profesional esto muchas veces no sea posible.

5.3.4 Uniformidad

El presente trabajo se realizó en respuesta a una necesidad apremiante de la UCR: uniformar la terminología sobre almacenes de datos que antes provenía de diferentes fuentes: textos educativos de España y México y textos extraídos de la Internet. Algunos de los ejemplos siguientes responden a esa necesidad.

Es evidente que los términos técnicos contribuyeron mucho a la complejidad de la traducción, en cuyo caso fue necesario consultar a la autora. Expresiones tales como “multivalued dimension”, “multivalued dependencies” o “multivalued attribute” en principio podrían indicar un patrón de uso, es decir, indicar que para todas las instancias la traducción sería dimensión/dependencia/atributo multivaluado, o bien, dimensión/dependencia/atributo de valores múltiples. No obstante, luego de corroborarlo con la autora, para “dependencia” o “atributo” el adjetivo correcto era “multivaluado”, no así para “dimensión” cuya traducción correcta era “dimensión de valores múltiples”. He aquí un nuevo ejemplo de la necesidad de contar con el aporte constante del autor al traducir un texto de esta naturaleza.

La expresión “data warehouse and OLAP applications” en algunas ocasiones también se describía en el texto fuente como “data warehouses and OLAP systems” (con una “s” al final de warehouse), lo cual al principio hizo pensar que se trataban de dos conceptos diferentes: el primero referido a un solo tipo de aplicación y el segundo referido a una aplicación y a un almacén. No obstante, al consultar con la autora ésta constató que, si bien ambas expresiones en inglés estaban correctas, para la traducción al español era necesario describirlas con una sola expresión, “sistemas de almacenes de datos y OLAP”.

5.3.5 Uso de sinónimos

Algunas palabras y expresiones requirieron modificar la traducción a los efectos de claridad como, por ejemplo, las palabras “organization” y “company” que en el original se usaban indistintamente. Si bien podría decirse que ambas son hasta cierto grado sinónimos, “company” tiene por lo general una connotación mercantil. Al conversar con la autora sobre esta diferenciación se decidió optar por el uso de compañía para uniformar la traducción, a pesar de que en una ocasión se optó por usar “empresa” para evitar cacofonía.

Asimismo, fue necesario evitar el uso de ciertos términos que, por solicitud expresa de la autora, podrían dar pie a confusiones. A modo de ilustración, la palabra “integrar” y cualquiera de sus derivaciones fue cambiada por un sinónimo, pues, según la autora, el concepto de integrar tiene un significado específico dentro de la terminología de almacenes de datos. Por ello, no era posible utilizar expresiones como “tal cosa está integrada por tal otra”. Un sinónimo opcional utilizado fue el verbo “formar” y sus derivaciones:

- De esta manera, las relaciones que ~~integran~~ forman jerarquías se conocen como relaciones padre-hijo.
- …pueden agregarse utilizando una jerarquía ~~integrada~~ formada por los niveles Profesión–Clase–Rama–Zona.

Otra palabra que a simple vista no debería ocasionar problemas en una traducción es “agregar” o “agregación”, que en el contexto de los almacenes de datos tiene un significado especial (véase la definición en el glosario) y la terminología de dicha área tiene ya expresiones fijas tales como “agregación de datos” o “agregar medidas” que tienen una connotación específica dentro del campo. Por ello fue importante asegurarse de que la palabra “agregar” en su definición de diccionario no fuera utilizada por error en un contexto equivocado. El término en inglés es “aggregate” o “aggregation”:

- Thus, the aggregation mechanism should be modified when a splitting level is reached in a roll-up operation.
- In order to aggregate measures correctly, a special aggregation procedure that uses this distributing factor must be implemented.

Para ampliar un poco más la idea, si en español había una frase u oración que podría aceptar la palabra “agregar” en su definición de diccionario, siempre se trató de buscar un sinónimo para evitar confusiones, digamos, “añadir”:

- Además, las llaves sustitutas no varían en el tiempo, de manera que dos entidades que tienen llaves sustitutas idénticas representan la misma identidad, lo cual permite añadir información histórica sin ambigüedades.

5.3.6 Regionalismos

Uno de los objetivos de esta traducción era adaptar el léxico al uso de las palabras que son más comunes en Costa Rica. Por ejemplo, “mobile phone”, que al principio se había traducido como

“teléfono móvil” o “móvil”, luego se cambió por “celular” que es el término empleado en el país para referirse al “mobile phone” o “cellular phone”. Además, el término “celular” es de uso común en Latinoamérica, mientras que “móvil” es más utilizado en España.

Con respecto a la diferencia entre “accesar” en lugar de “acceder” cuando se refería a información (“to access the information”), fue necesario aclarar a la autora la diferencia, tal y como lo explica el Diccionario Panhispánico de Dudas que, de hecho, da un ejemplo de un texto procedente del periódico “La Nación” de Costa Rica:

> En gran parte de América, especialmente en México y el área centroamericana, se ha extendido el uso del verbo transitivo accesar (del ingl. to access) con el sentido de ‘acceder [a información o datos contenidos en un sistema informático]’: «El lector puede accesarla [la información] mediante un buscador que funciona con palabras claves» (Nación [C. Rica] 7.4.97). Se trata de un anglicismo innecesario, que debe sustituirse por el verbo intransitivo español acceder: El lector puede acceder a la información mediante...

Evidentemente, la autora determinó que, a pesar de ser un regionalismo aceptado en el país, era importante que en la traducción se utilizara el término correcto de conformidad con lo indica la Real Academia.

5.3.7 Limitaciones de la Internet para la búsqueda de términos

Si bien la mayoría de los términos técnicos pudieron encontrarse en diferentes textos de la Internet (verificados todos ellos por la autora) sí es importante notar que, a pesar de realizar diversas búsquedas avanzadas, hubo dos términos que dieron qué pensar a la autora, quien necesitó algún tiempo tomar una decisión respecto de los dos. Los dos términos que requirieron la atención especial de la autora fueron “non-covering hierarchies” y “non-onto hierarchies”. En el primer caso, encontré un sitio virtual en España que las denominaba “jerarquías no cubiertas”, pero al inicio la autora no estaba satisfecha con dicho término, a pesar de que la fuente era una tesis de la Universidad Politécnica de Valencia. No obstante, al no encontrar un término más concreto terminó por aceptar el uso de “no cubiertas” en la versión final. El término queda abierto a modificación en el futuro, en caso de que la autora encuentre una palabra (o frase) que la describa mejor. La segunda palabra también había quedado pendiente hasta el momento de preparar la versión definitiva por falta de un término más apropiado: “non-onto”. Al principio se

pensó en traducirlo como jerarquías “sin hojas” porque, según la autora, era lo que mejor lo describía, pero a última hora desistió de la idea y decidió dejarlo en inglés. Es decir, este fue el único término no traducido.

5.3.8 Errores en el original

Si bien los errores en el original fueron muy pocos es importante mencionar aquí algo que es muy común en un proceso de traducción y es que los autores no son infalibles y se les pasan errores, aunque el libro o publicación haya pasado por las manos de revisores. En el caso de esta traducción los errores se identificaron cuando la autora dio la lectura final al documento y a la hora de ingresar los cambios a la traducción verifiqué el original, puesto que como traductora quise corroborar si el error que ella me marcaba era del original o de la traducción. Un ejemplo fue el término “Jerarquía Calendario” que originalmente decía “Jerarquía Tiempo”. Quizás vale la pena mencionar que la autora realizó la lectura final en mi presencia y así yo pude ir digitando y verificando contra el original simultáneamente. Cuando le mostré a la autora el original (que decía “Time Hierarchy”), ella confirmó que, en definitiva, se trataba de un error en el original. Otro caso fue la expresión “En vista de que el saldo depende del monto específico…” que la autora corrigió por “En vista de que el saldo depende de la cuenta específica…” y que, al verificar el original, se verificó que por error decía “amount” en lugar de “account”.

Quizá también valga la pena mencionar brevemente que al empezar la traducción del capítulo se contó tanto con el libro publicado como con un archivo en formato pdf enviado por la autora para facilitar la traducción en pantalla (en dos ventanas). No obstante, luego de haber traducido algunas páginas me di cuenta de que el pdf no coincidía con el contenido del libro y luego de consultar con la autora me explicó que me había entregado una revisión equivocada. Me parece que es importante mencionarlo porque, si bien creo que no es un problema común, sí es prueba de que en un trabajo de traducción puede ocurrir cualquier contratiempo, especialmente cuando se trabaja con tantas versiones (y este trabajo no es la excepción). En este caso el trabajo realizado hasta ese momento no se vio afectado por el error, pero cuántas veces ese no es el caso; empezar una traducción y luego enterarse de que el trabajo que uno ha hecho no sirvió de nada.

6 Estilo

El hecho de que el inglés tenga preeminencia sobre los otros idiomas cuando se trata de traducción técnica no justifica que los textos traducidos tengan que copiar los términos y, menos aún, las estructuras gramaticales de ese idioma. Si bien fue a partir de los años cincuenta que empezó a surgir un gran número de obras científicas escritas en inglés que se llegaron a equiparar en número (y luego sobrepasar) al alemán y al francés, hoy en día lenguas como el español cada vez tienen mayor preeminencia en respuesta al fenómeno de la globalización y a la creciente inmigración acentuada por la crisis mundial que afecta a la mayoría de las naciones. Por ejemplo, de acuerdo con foreigntranslations.com, sólo en los Estados Unidos hay más de 50 millones de personas que hablan ese idioma y el español es la segunda lengua más hablada en el país; asimismo, el 60% de los estudiantes estadounidenses lo escogen como lengua extranjera en la universidad, lo cual hace pensar que el idioma español seguirá esa tendencia de crecimiento.

Es cierto que el traductor técnico debe documentarse y entender los conceptos básicos del tema a traducir, pero ello no debe limitarlo a la simple traducción de términos sino que también debe conocer las reglas ortotipográficas[4] del español sin dejarse influir por el uso del inglés, ya que un error de este tipo podría incluso cambiar el significado de una expresión. Por ello es importante repasar aquí algunos de los ajustes estilísticos efectuados durante la preparación de este trabajo.

6.1.1 Lengua materna de los autores de libro

Uno de aspectos interesantes de este proyecto fue el hecho de que los autores del texto fuente redactaron en inglés, a pesar de que sus lenguas maternas eran el polaco y el francés. Este fenómeno es cada vez más común, no sólo en el área de las ciencias sino en muchas otras conforme la globalización en el campo de la educación y otros ámbitos de la investigación son partícipes de iniciativas de colaboración e intercambio entre expertos procedentes de diversas culturas que muchas veces manejan dos o más idiomas. Por ello, consideré interesante mencionar

---

[4] La ortografía es, según el Diccionario de la Real Academia Española, el conjunto de normas que regulan la escritura de una lengua, y la ortotipografía es el conjunto de usos y convenciones particulares por las que se rige en cada lengua la escritura mediante signos tipográficos. Javier Bezos (2007) define la *tipografía* como el arte de crear y combinar tipos, es decir, letras de imprenta, para producir libros, revistas, folletos, etc., con el objetivo de facilitar su lectura y que el contenido se transmita de forma eficaz.

aquí, por ejemplo, la revisión hecha por la autora al primer borrador en la cual corrigió el término técnico “atributos llave” por atributos llaves” y “niveles raíz” por “niveles raíces”. En este caso fue necesario explicarle a la autora (quien, vale decir, tiene un dominio excelente del idioma español) la norma ortográfica referida a las aposiciones de sustantivos según la cual debe pluralizarse sólo el primer componente: “niños prodigio” o “palabras clave”.

6.1.2 Uso de cursiva versus negrita

De acuerdo con Gonzalo Claros (2008):

> En inglés científico, aunque es raro, puede emplearse la cursiva para dar énfasis a alguna palabra. Este uso está recogido en español, aunque, a veces, el énfasis se puede expresar bien con comillas o bien con negritas (poco recomendable) […] En resumen, en inglés y en español se usan las cursivas para dar énfasis. Para esto también se pueden utilizar las negritas, pero estilísticamente se prefieren las cursivas.

En el texto original en inglés del capítulo traducido los autores utilizan las negritas cada vez que desean resaltar un término. En español se cambió a cursivas cumpliendo con esta preferencia estilística del idioma español. A continuación se muestra un ejemplo de este ajuste:

- Generalized hierarchies include a special case commonly referred to as **noncovering hierarchies** or **ragged hierarchies**
- Dentro de las jerarquías generalizadas existe un caso especial que generalmente se conoce como *jerarquías no cubiertas* o *jerarquías desiguales.*

Por preferencia estilística escogí cursiva y no comillas, pues las comillas, según mi apreciación, interrumpen un poco la lectura mientras que la cursiva pasa un poco más desapercibida sin quitarle el énfasis que se quiere dar a la expresión.

6.1.3 Fusión de oraciones

Es muy común que en español se unan oraciones para transmitir el mensaje más claramente. El idioma inglés tiende a utilizar varias oraciones cortas para expresar una; no obstante, el español puede expresar las mismas ideas en una sola frase, según puede observarse aquí:

- A fact relationship may contain attributes commonly called measures. These contain data (usually numerical) that is analyzed using the various perspectives represented by the dimensions.
- Una relación de hechos podría tener atributos comúnmente denominados medidas, las cuales contienen datos (por lo general numéricos) que se analizan utilizando las diversas perspectivas representadas por las dimensiones.

6.1.4 Repeticiones

Como es bien sabido, en el idioma inglés las repeticiones dentro de un mismo párrafo son muy comunes, lo cual requiere encontrar un sustituto para el español y así evitar la redundancia y mejorar la fluidez de la lectura. Las oraciones en inglés por lo general son cortas y directas, pero en español esto ocasiona que el texto suene cortado y dispar. Un ejemplo sería el siguiente:

- We use the term "nonstrict hierarchy" to denote an acyclic classification graph. We use this term for several reasons.
- Utilizamos el término *jerarquía no estricta* para denotar un gráfico acíclico de clasificación [104] por varias razones.

La expresión "for example" del inglés tiene otra opción corta, "for instance", lo cual no ocurre en español, y de ahí que el problema de repetición de "por ejemplo" sea más común en español. El uso de expresiones como "a modo de ejemplo" o "a manera de ilustración" no se utilizaron como sinónimos ya que, por ser un texto educativo o ilustrativo, interfería mucho con la lectura.

6.1.5 Puntuación

Es de todos sabido que la puntuación en ambos idiomas tiende a variar a la hora de traducir con el fin de facilitar la lectura. Muchas veces esta decisión queda en manos del traductor, quien tiene la labor de asegurarse de que su texto traducido no se preste a confusiones. Ejemplo de ello es la siguiente frase:

- In this section, we define the MultiDim model, a conceptual multidimensional model.

traducida al español como

- En esta sección definimos el modelo MultiDim: un modelo multidimensional conceptual.

Otro ejemplo donde no sólo se modificó la puntuación sino también se eliminó la abreviatura "i.e." (ver también "Omisiones") por considerarse innecesaria es el siguiente:

- In a strict hierarchy, this product can only belong to one category, i.e., “phone”. In a nonstrict hierarchy it may be considered as belonging to three categories, i.e., “phone”, “PDA”, and “MP3 player”.
- En una jerarquía estricta, este producto puede pertenecer únicamente a una categoría: “teléfono”. En una jerarquía no estricta podría pertenecer a tres categorías: “teléfono”, “PDA” y “reproductor de MP3”.

Un aspecto interesante de la puntuación es el uso de la raya (—) que quizás es más común en el idioma inglés. Si bien en el texto fuente no hubo instancias en que se utilizara, en la traducción se utilizó unas cuantas veces para dar una claridad “visual” al texto. He aquí un ejemplo en el que el uso de muchas comas podría obstaculizar la lectura fluida del texto:

- Note that in the mobile phone example, distribution of the sales measure between three categories according to previously specified percentages, for example 70% to “phone”, 20% to “PDA”, and 10% to “MP3”, may not give meaningful information to users.

Considero que el texto quedaría mejor si lo traducimos con la siguiente puntuación:

- Vale notar que, en el ejemplo del celular, la distribución de la medida de ventas entre tres categorías sobre la base de porcentajes previamente estipulados —por ejemplo, 70% para “teléfono”, 20% para “PDA” y 10% para “MP3”— quizá no brinde a los usuarios información significativa.

Por cierto, otra opción hubiera sido utilizar paréntesis en lugar de raya; no obstante, mi opinión es que la raya da un “respiro” a la lectura sin interrumpirla mientras que el paréntesis podría pasar desapercibido. Martínez de Souza (2001:153) explica, a propósito de este signo, que:

> Cuando se emplea en su función parentética, tiene el mismo empleo y la misma grafía que el paréntesis: va precedido de blanco y seguido de blanco (el propio de la línea en cada caso) o de un signo de puntuación.

6.1.6 Omisiones

El tema de omisiones está muy ligado al de repeticiones. En algunos casos debemos recurrir a la omisión de palabras que no agregan nada a la traducción y también para evitar repeticiones. Un buen ejemplo de ello se encuentra en la oración:

- The lower path (i.e., the path containing the Profession and Class levels) corresponds to the Person subtype, while the upper path (i.e., that containing the Type and Sector levels) indicates the Company subtype.

Aquí, la abreviatura "i.e." podría eliminarse, así como el segundo "path" para que la lectura sea más fluida, según se muestra a continuación:

- La ruta inferior (la que contiene los niveles Profesión y Clase) corresponde al subtipo Persona, mientras que la superior (la que contiene los niveles Tipo y Sector) indica el subtipo Compañía.

Otro ejemplo interesante es el que se presenta en la frase siguiente, el cual además muestra que en la traducción siempre hay cabida para "mejorar el original":

- Since the balance depends on the specific account and the time when it is considered, we have the situation that the account and the time determine the balance.

La traducción de la frase quedó así:

- En vista de que el saldo depende del monto específico y del momento en el que se examina, la cuenta y el tiempo son los que determinan el saldo.

Obviamente, en este caso, no es necesario añadir la frase "we have the situation that" que no aporta nada al sentido de la oración y nos permite ajustar el texto de manera que se lea con fluidez.

6.1.7 Adiciones

Así como omitimos palabras, también podemos agregarlas para redondear una idea. Es muy común que en inglés se omitan ciertas estructuras gramaticales, a menudo verbos que, en español, necesitarían una adición pues al leerse el texto parecería que no suena natural. Veamos un ejemplo extraído del texto:

- Further, the distinction between splitting and joining levels in generalized hierarchies is important for correct measure aggregation during roll-up and drill-down operations.
- Adicionalmente, en las jerarquías generalizadas es importante hacer la distinción entre niveles de separación y unión con el fin de obtener una correcta agregación de medidas durante las operaciones de navegación ascendente y descendente.

6.1.8 Cacofonía

Una de las dificultades con la que se enfrenta todo traductor es poder evitar la cacofonía en su redacción. La mayoría de las veces es posible evitarla como, por ejemplo, aquí:

- Las jerarquías no estrictas ~~presentan~~ acarrean el problema del conteo doble de las medidas cuando una operación…

6.1.9 Orden de oraciones

Es de conocimiento popular entre los que aprenden la lengua inglesa que el orden de los componentes de una oración cambian con respecto al orden en español. En este proyecto no hubo excepciones. Veamos a continuación un ejemplo.

- The choice of an appropriate distributing factor is important in order to avoid approximate results.

Una frase tan sencilla como esta podría pasar por desapercibida y ser traducida en el mismo orden en que el inglés la presenta. No obstante, lo mejor sería adaptar el orden a lo que naturalmente se diría en español:

- Es importante seleccionar un factor de distribución adecuado para evitar que se obtengan resultados aproximados.

7 Conclusiones

El campo de la traducción técnica orientada al tema de almacenes de datos tiene buenos prospectos de evolución conforme el tema se hace cada vez más relevante en el ámbito comercial y conforme avanza la tecnología y surgen nuevos programas informáticos y herramientas que ayuden al experto a analizar la información y generar informes veraces. Específicamente la importancia del libro utilizado para este trabajo ha sido puesta en evidencia gracias al respaldo recibido del padre de los almacenes de datos, William Inmon.

Sin duda la traducción técnica es una labor que requiere una gran cantidad de tiempo y dedicación al tema en cuestión. Es evidente que un traductor técnico no puede tener los conocimientos y habilidades de los expertos en la materia, pero sí es posible presentar un producto final de calidad valiéndose no solamente de los diccionarios disponibles sino también

de las herramientas de búsqueda avanzada en la Internet y recurriendo constantemente a la retroalimentación del especialista para la verificación de términos y conceptos específicos. En el caso de la Internet se constató que, en el tema de almacenes de datos, es difícil encontrar una fuente que utilice la terminología de manera universal; cada país y cada especialista usa su propio lenguaje, aunque algunos términos sí tienen un uso más generalizado. Esto conlleva el problema de fiabilidad de la información por lo que, de nuevo, la intervención y revisión del experto es de suma importancia.

Otro aspecto importante al realizar una traducción técnica es tener la capacidad de entender o estar con la disponibilidad de aprender un programa informático nuevo. En este proyecto fue necesario aprender el uso del programa informático Visio para editar las figuras del capítulo. Pensar que se trata simplemente de entrar en el programa y cambiar el texto es incorrecto pues, como ya se dijo, el texto en español ocupa un mayor espacio que el texto en inglés; por lo tanto, es necesario saber cómo manipular las figuras de manera que el texto calce perfectamente. Es un ejercicio lento que requiere gran paciencia y concentración ya que, como en todo programa nuevo que se aprende, al principio se cometen errores que a veces nos fuerzan a empezar la labor desde un principio e incluso, mientras se logra perfeccionar su uso, en algunas ocasiones perdemos, por error involuntario, información que ya habíamos incorporado. El hecho de poder tomar un curso de Visual Basic también me ayudó a tener una comprensión más general del proceso. Verdaderamente, el trabajo de traducción conlleva mucho más que la simple traducción de palabras.

Por medio de este trabajo se corroboró que los mitos que Byrne menciona son, como él lo apunta, totalmente falsos, pues pude constatar lo siguiente: la terminología ayuda a transmitir información pero sólo representa un porcentaje mínimo del texto técnico; el estilo sí es relevante, tanto como en cualquier otro tipo de traducción; los traductores técnicos usan la creatividad para comunicar el mensaje satisfactoriamente; no es necesario ser experto en la materia, lo que se requiere es poseer excelentes destrezas de investigación; la información que se presenta debe ser precisa, pero también presentarse en forma correcta y completa de manera que pueda utilizarse de manera apropiada y eficaz. Otro mito importante que menciona Byrne es la creencia de que una traducción técnica debe ser casi exclusivamente literal. Mediante este trabajo se corroboró

que también es necesario poseer un sólido conocimiento del idioma y de las estrategias y decisiones de traducción que permitan ofrecer un producto final de alta calidad.

Por otra parte, el papel del traductor técnico consiste en asegurarse de que la persona que en última instancia lee el texto pueda hacerlo con relativa facilidad y que, cualquiera que sea la tarea que el lector debe llevar a cabo, ésta le resulte más fácil tras la lectura del texto. Esto es muy cierto en el caso del presente trabajo donde el objetivo principal es entregar un producto final que pueda ser utilizado por los estudiantes y profesores de la UCR y que sirva de base para futuras traducciones o publicaciones. Por ello es importante que una traducción de este tipo sea realizada por traductores profesionales que sepan cómo utilizar las herramientas para hacerlo y que tengan una comunicación constante con el o los expertos que han redactado el documento original o que sean especialistas en la materia.

Con respecto a cómo enmarcar una traducción técnica es importante mencionar que la propuesta de Byrne de encuadrarla dentro de la teoría de Skopos me parece acertada, por tanto se trata de un "servicio comunicativo orientado al lector", es decir, el producto final sirve para que el lector ponga en práctica con facilidad, oportunidad y precisión las instrucciones, pautas o descripciones que indica el texto. De no ser así, el propósito de la traducción estaría perdido, pues es precisamente el propósito (Skopos) del texto meta el que determina los métodos y estrategias de traducción, de acuerdo con las necesidades del cliente (en nuestro caso, los alumnos y profesores de la UCR).

Por último, pero no menos importante, la elaboración del glosario fue quizás el elemento fundamental para la traducción del capítulo 3, pues requirió muchas horas de investigación y consulta constante con la especialista. La ventaja de disponer de un glosario antes de empezar una traducción es obvia: la traducción se hace más rápida y fluidamente e incluso se llega al punto de que los términos se asimilan y la consulta constante al glosario (e incluso al especialista) ya no es tan periódica. Un aspecto importante del glosario es el hecho de que cada término viene acompañado de su definición. De esta manera, el traductor también tiene la posibilidad de comprender un poco más en detalle lo que significa determinado término dentro de cierto contexto. El otro aspecto importante de contar con un glosario es la posibilidad de seguir agregando términos que se hayan quedado por fuera durante la redacción del texto

original. Al respecto, considero interesante mencionar que el hecho de no ser experto en la materia ofrece una oportunidad de retroalimentación importante para los escritores del texto original que quizás no se percatan de ciertos términos que deben ser incluidos o de algunas definiciones que podrían mejorarse en el original, e incluso de faltas ortográficas o mecanográficas del mismo. Esto puede ser útil si los autores piensan imprimir una nueva versión de su publicación en el futuro.

En definitiva, el presente trabajo me ha permitido tener una mejor comprensión de todo el proceso de traducción y lo que conlleva la preparación de una traducción técnica de este tipo, desde la recopilación terminológica, la investigación exhaustiva, el contacto directo con el experto, la traducción propiamente dicha e incluso la necesidad de aprender acerca del uso de programas informáticos necesarios para entregar un producto final de alta calidad. La Dra. Malinowski se mostró plenamente satisfecha con la traducción de este importante capítulo y del glosario, los cuales ya están siendo utilizados por los estudiantes y profesores de la Universidad de Costa Rica.

8 Bibliografía

AVELING, Harry (2002) “Mistakes” in Translation A Functionalist Approach. <http://www.soas.ac.uk/literatures/satranslations/Aveling.pdf> [Consulta: 2 de julio de 2010]

BAKER, Mona: *Routledge encyclopedia of translation studies*. Routledge: London. 2001.

BEYE NETWORK: *Book review: Advanced data warehouse design*. <http://www.b-eye-network.in/view/9959> [Consulta: 2 de julio de 2010].

BEZOS, J. (2007): *Tex y tipografía*. <www.texytipografia.com/typo.html> [Consulta: 26/6/2010].

BYRNE, Jody: *Technical Translation. Usability strategies for translating technical documentation.* The Netherlands: Springer, 2006.

CABRÉ, María Teresa: *Terminology. Theory, methods and applications*, Amsterdam: John Benjamins Publishing Company, 1992.

CORPORATE INFORMATION FACTORY: About Bill. http://www.inmoncif.com/about/. [Consulta: 30 de junio de 2010].

GONÇALVES BARBOSA, Heloisa: *Tradução, mercado e profissão no Brasil*. Universidade Federal do Rio de Janeiro (UFRJ) http://www.confluencias.net/n3/barbosa.pdf [Consulta: 30 de junio de 2010].

GENTZLER, Edwin: *Contemporary translation theories*. Revised Second Edition. Cromwell Press Limited: Great Britain 2001.

GUIDÈRE, Mathieu: *Introduction à la traductologie: Penser la traduction: hier, aujourd’hui, demain.* Groupe De Boeck: Bruxelles. 2008.

HATIM, Basil: *Teaching and Researching Translation*. Harlow: Pearson Education. 2001.

HERVEY, Sándor; Higgins, Ian y Haywood, Louise M.: *Thinking Spanish Translation. A course in translation method: Spanish to English*. Routledge: London. 1995.

HÖNIG, Hans: *Positions, Power and Practice: Functionalist Approaches and Translation Quality Assessment*. <http://pdfserve.informaworld.com/240492__921563300.pdf> 1998.

KAGEURA, Kyo: *The Dynamics of Terminology, A descriptive theory of term formation and terminological growth*, 5 vols., Amterdam: John Benjamins Publishing Company, 2002.

MALINOWSKI, Elzbieta. Consulta realizada vía correo electrónico del 3 de julio de 2010.

NING, Zhao. *Skopos in Practice: Building an Appealing Brand Image in the Translation of Soft News.* < *http://accurapid.com/journal/46skopos.htm*> [Consulta: 30 de junio de 2010]

NORD, Christiane: Translating as a Purposeful Activity. Manchester: St.Jerome. 1997

PALUMBO, Giuseppe: *Key terms in translation studies*. New York: Continuum. 2009.

Pérez Hernández, Chantal: *Explotación de los córpora textuales informatizados para la creación de bases de datos terminológicas basadas en el conocimiento* < http://elies.rediris.es/elies18/index.html> [Consulta: 26/7/2010]

POLCHLOPEK, Silvana y de Abreu Alo, Michelle: *Tradução técnica: Armadilhas e desafios*. Revista brasileira de tradutores. No. 19, 2009.

REAL ACADEMIA ESPAÑOLA (2001): Diccionario de la lengua española, 21.ª ed. Madrid: Espasa Calpe. <http://buscon.rae.es/diccionario/drae.htm> [Consulta: 26/6/2010].

RED IRIS. Universidad Autónoma de Barcelona: *Normalización terminológica: esfuerzos de estandarización e instituciones normalizadoras*. <http://elies.rediris.es/elies18/321.html> [Consulta: 04/01/2010].

REY, Alain: *Essays on Terminology*, 9 vols. Amsterdam: John Benjamins Publishing Company, 1995.

ROMÃO, Wesley; Dias, Maria Madalena; Mattos, Mauro M.; Todesco, José Leomar; Pacheco, Roberto C.: *Pesquisas futuras em data warehouse* <http://www.inf.furb.br/~mattos/arquivos/publicacoes/datawarehouse.pdf > [Consulta: 5 de junio de 2010].

SAGER, Juan C.: *A practical course in terminology processing*, Amsterdam: John Benjamins Publishing Company, 1990.

SANTIESTEVAN, Stina: *Use of the Spanish Language in the United States: Trends, Challenges, and Opportunities*. <http://www.ericdigests.org/pre-9221/spanish.htm> [Consulta: 30 de junio de 2010].

SOLANA, Maité: *Traducción, adaptación y fidelidad*. <http://medtrad.org/panacea/IndiceGeneral/n13-14_entremes_solana.pdf> s.a. [Consulta: 2 de julio de 2010].

TURBAN, Efraim; Aronson, Jay E.: *Decision support systems and intelligent systems*. New Jersey: Prentice Hall, 1998.

VERMEER, Hans J.: Skopos and Commission in Translational Action. En Chesterman, A. (ed.). Readings in translation theory. 1989

VELASCO, Roberto. <http://www.rhernando.net/modules/tutorials/doc/bd/dw.html> s.a. [Consulta: 3 de julio de 2010].

WIKIPEDIA. La Enciclopedia Libre. <http://es.wikipedia.org/wiki/Almac%C3%A9n_de_datos> [Consulta: 2 de julio de 2010].

—http://en.wikipedia.org/wiki/Data_warehouse [Consulta: 3 de julio de 2010].

Bibliografía del Glosario

ANDRADE, Danny, *et al.*: *Análisis y diseño del modelo de información del sector eléctrico ecuatoriano relacionado con la distribución y comercialización de energía eléctrica*. <http://www.dspace.espol.edu.ec> [Consulta 18-12-2009].

ANGOITIA ESPINOZA, Itziar: *Data warehouse para la gestión de lista de espera sanitaria*, Facultad de Informática, Universidad Politécnica de Madrid. <http://oa.upm.es/1095/1/PFC_ITZIAR_ANGOITIA_ESPINOSA.pdf> [Consulta: 29-12-2009]

Autentia (adictosaltrabajo.com): *Todo lo que querías saber sobre Datawarehouse IV*, (tutorial en línea por Christhian Kirs Herrera Basurto), Quito, 2007. http://www.adictosaltrabajo.com/tutoriales/pdfs/datawarehouse.pdf [Consulta 22-12-2009].

Automation and IT, Online dictionary Spanish-English <http://www.mijnwoordenboek.nl/EN/theme/AU/ES/EN/N/1.html> [Consulta 22-12-2009].

BATINI, Carlo et al.: *Diseño conceptual de bases de datos*, Delaware, EEUU: Addison-Wesley Publishing Company, 1992 <http://books.google.com/books?id=-DP-0puz338C&printsec=frontcover&dq=Dise%C3%B1o+conceptual+de+bases+de+datos+By+Carlo+Batini&cd=1#v=onepage&q=ra%C3%ADz&f=false> [Consulta: 20-12-2009].

blobgle.com: *Quien comprara a Microstrategy?*, <http://blobgle.com/blog/?cat=12> [Consulta: 23-12-2009].

CABRÉ, María Teresa: *Terminology. Theory, methods and applications*, Amsterdam: John Benjamins Publishing Company, 1992.

Centro de Altos Estudios en Tecnología Informática: *Aplicando Restricciones en un Datawarehouse Temporal Utilizando UML/OCL*, Facultad de Tecnología Informática, Universidad Abierta Interamericana, Argentina, <http://caeti.uai.edu.ar/archivos/194_APLICANDO__RESTRICCIONES__DWTEMPORAL_CACIC2003.PDF> [Consulta: 19-12-2009].

Data Mining Institute: *Data Mining con SAS*, <http://www.estadistico.com/kurs.html?DMS> [Consulta: 18-12-2009].

DE LA ROSA PIÑEIRO, Antonio: Sistemas de bases de datos relacionales orientados a objetos e interfaces vrml para el www, en *El Profesional de la Información*, Revista internacional científica y profesional, marzo 2008, <http://www.elprofesionaldelainformacion.com/contenidos/1998/marzo/sistemas_de_bases_de_datos_relacionales_orientados_a_objetos_e_interfaces_vrml_para_el_www.html> [Consulta 20-12-2009].

Departamento de Lenguajes y Sistemas Informáticos II. Grupo de Investigación Kybele. Universidad Rey Juan Carlos. *Datawarehouse. Inteligencia de Negocios*. Madrid, 2007. <http://www.dlsi2.etsii.urjc.es/> [Consulta: 20-12-2009].

DÍAZ VILLANUEVA, Wladimiro: *Bases de datos objeto-relacionales*; Curso 2001-2002, Departamento de Informática, Universidad de Valencia <http://informatica.uv.es/iiguia/DBD/Practicas/boletin_1.pdf> [Consulta:19-12-2009].

GUTIÉRREZ, Mariella: *El Rol de las Bases de Datos Espaciales en una Infraestructura de Datos*. Ponencia durante la Conferencia del GSDI-9, 6 al 10 de noviembre de 2006, Santiago, Chile, <http://www.cp-idea.org/nuevoSitio/documentos/tecnologia/Rol%20de%20las%20bases%20de%20datos%20espaciales%20en%20una%20IDE.pdf> [Consulta: 19-12-2009].

Hewlett-Packard Development Company: *Glosario: Guía de ayuda para la funcionalidad OL* de las tarjetas de interface HP-UX 11i v2*, <http://docs.hp.com/es/5992-1928/go01.html> [Consulta 22-12-2009].

—*Glosario: HP Capacity Advisor versión 4.0: Guía del usuario* <http://docs.hp.com/es/T8670-90004/go01.html> [Consulta 22-12-2009].

IBM: *Glosario de IBM (IBM® DB2 Universal Database™)*, <ftp://public.dhe.ibm.com/ps/products/db2/info/vr8/pdf/letter/nlv/db2t0z80.pdf> [Consulta: 18-12-2009].

intraEmprendedor.com: *Qué es una Slowly Changing Dimension (SCD)*, <http://www.intraemprendedor.com/2009/11/13/que-es-una-slowly-changing-dimension-scd-what-the-hell-2/> [Consulta: 18-12-2009].

JOYANES AGUILAR, Luis: *Programación orientada a objetos*, Facultad de Informática, Universidad Pontificia de Salamanca, Madrid: McGraw Hill, 1996. http://api.ning.com/files/UlmjdymDhjZH*vLu230duux*CjPZcH2yYHrFxZ8vRbrPWi2r2BaLiw5P*njMaG4ACrKBra4lQ7zvPGcc-f24hWUEdyp8g1Yr/ProgramacionOrientadaaObjetosLuisJoyanesAguilar.pdf [Consulta 18-12-2009].

Kageura, Kyo: *The Dynamics of Terminology, A descriptive theory of term formation and terminological growth*, 5 vols., Amterdam: John Benjamins Publishing Company, 2002.

KROENKE M. David: *Procesamiento de bases de datos: fundamentos, diseño e implementación*: Pearson Prentice Hall, 2005.

MANEIRO, Mariela Yanina: *Minería de datos*, Tesis de Licenciatura, Facultad de Ciencias Exactas, Naturales y Agrimensura, Universidad Nacional del Nordeste, Argentina: 2008. <http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MineriaDatosYany2008.pdf> [Consulta 22-12-2009].

MARRERO ANTUNEZ, Ivette: *La inteligencia de negocios desde la perspectiva cubana: retos y tendencias*, Havana: Consultoría Biomundi. Instituto de Información Científica y Tecnológica (IICT), 2008. <http://www.stratebi.es/todobi/may08/BI%20en%20Cuba.pdf> [Consulta: 18-12-2009].

Microsoft Development Network (MSDN): *Glosario de términos, SQL Server 2005*, <http://msdn.microsoft.com/es-es/library/ms165911(SQL.90).aspx> [Consulta: 20-12-2009].

MOLINA FÉLIX, Luis Carlos: *Data mining: torturando a los datos hasta que confiesen*, Universitat Oberta de Catalunya, 2002 <http://www.uoc.edu/web/esp/art/uoc/molina1102/molina1102.html> [Consulta 23-12-2009].

Monografías.com S.A.: *Introducción a Borland C++ 5.0.*, <http://www.monografias.com/trabajos56/borland-c/borland-c2.shtml> [Consulta: 19-12-2009].

MUÑIZ CASTRO, Emilio-Germán: *Routledge English-Spanish Dictionary of Telecommunications; Routledge Spanish Dictionary of Telecommunications/Diccionario Inglés de Telecomunicaciones: Spanish-English/English-Spanish (Routledge Bilingual Specialist Dictionaries)*, Routledge, 1998 <http://books.google.com/books?id=cr0I051BXlUC&pg=PA302&lpg=PA302&dq=%22almac%C3%A9n+de+datos%22+partici%C3%B3n+(partitioning)&source=bl&ots=SUgDE5I76K&sig=rKrS5OwIZ6jZbVJql9115YjL5UI&hl=en&ei=pf8uS-njKMnPlAet-6WkBw&sa=X&oi=book_result&ct=result&resnum=2&ved=0CA0Q6AEwAQ#v=onepage&q=ATTRIBUTE&f=false> [Consulta 20-12-2009].

NADER, Javier: *Sistema de Apoyo Gerencial Universitario*, Tesis de Magister en Ingeniería del Software, Instituto Tecnológico de Buenos Aires, Argentina: 2002. <http://www.itba.edu.ar/archivos/secciones/nader-tesisdemagister.pdf> [Consulta: 21-12-2009].

PELANÉS GRAMAJE, Mª Carmen: *Una aproximación tecnológica al Desarrollo de Flujos de trabajo*, Tesis doctoral. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2002. <http://www.dsic.upv.es/docs/bib-dig/tesis/etd-10272003-001444/Tesis.pdf> [Consulta: 19-12-2009].

Polo de Desarrollo Educativo Renovador Po.D.E.R.: *La Tecnología y el Procesamiento Electrónico de Datos. Los Sistemas de Información General. La Tecnología y el Procesamiento Electrónico de Datos.* (tutorial en línea por Edgardo Sajón). Argentina. <http://www.educacionparapoder.com.ar/main.html> [Consulta: 20-12-2009].

Programa de las Naciones Unidas para el Desarrollo (PNUD): *Elaboración y puesta en funcionamiento del sistema de información SISBEN 2006*, Bogotá: 2005. <http://licitaciones.pnud.org.co/Docs/50791/ANEXO4.PDF> [Consulta: 19-12-2009].

Programación en castellano: *El lenguaje de programación* (tutorial en línea por José Antonio González Seco), España, 2002. http://www.programacion.com/ [Consulta: 18-12-2009].

PROZ.com, The translation workplace: *Solución de minería de datos*, <http://www.proz.com/kudoz/english_to_spanish/business_commerce_general/1168625-data_mining_solution_en_este_contexto_dir%C3%ADamos_investigador.html> [Consulta: 23-12-2009].

REAL ACADEMIA ESPAÑOLA, Diccionario de la Lengua Española, XXII edición en línea, <http://www.rae.es/rae.html> [Consulta: 23-12-2009].

Rey, Alain: *Essays on Terminology*, 9 vols. Amsterdam: John Benjamins Publishing Company, 1995.

ROB, Peter y Coronel, Carlos: *Sistemas de bases de datos: diseño, implementación y administración*, México: 2004 <http://books.google.es/books?id=B_UVi51RDY4C&sitesec=buy&source=gbs_navlinks_s> [Consulta: 22-12-2009]

Sager, Juan C.: *A practical course in terminology processing,* Amsterdam: John Benjamins Publishing Company, 1990.

TARRUELA, Josep: *Creando el próximo Data Warehouse: Integración y Calidad de Datos, (Sesión 1: Fundamentos del DWH)*, Aules d'Empresa 2010. Facultad de Informática de Barcelona (FIB). <http://www.aulesempresa.upc.edu/anteriors/2008/programes/PWD_DWH_S1_FundamentosDWH.pdf> [Consulta: 22-12-2009].

TRUJILLO, Carlos Carlos, et al.: *Apuntes sobre Almacenes de datos, Bases de Datos Multidimensionales y Herramientas OLAP,* Bubok Publishing S.L., 2009.

U-Cursos. Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile: *Inteligencia de Negocios: Análisis dimensional*, <www.ucursos.cl/ingenieria/2007/1/CC60K/1/material_docente/objeto/122713> [Consulta 22-12-2009].

Universidad de Argentina (UBA). *Sistemas operativos: Glosario del Departamento de Computación,* Facultad de Ciencias Exactas y Naturales, Actualizado a marzo de 2009, <http://www-2.dc.uba.ar/materias/so/datos/glosario.pdf> [Consulta: 19-12-2009].

Wikipedia, La Enciclopedia Libre (En línea): *Almacén de Datos* <http://es.wikipedia.org> [Consulta: 19-12-2009].

—*Base de datos multidimensional,* [Consulta: 22-12-2009].

—*Bases de Datos Temporal* [Consulta: 18-12-2009].

—*Herramienta CASE,* [Consulta: 19-12-2009].

—*Inteligencia Empresarial,* [Consulta: 18-12-2009].

—*Modelo entidad-relación,* [Consulta: 20-12-2009].

—*Cubo OLAP,* [Consulta: 21-12-2009].

—*Minería de datos* [Consulta: 21-12-2009].

—*Sistema de Información Geográfica (SIG),* [Consulta: 23-12-2009].

—*Object Management Group (OMG),* [Consulta: 18-12-2009].

Wordreference.com, <http://forum.wordreference.com/showthread.php?t=112090> [Consulta: 22-12-2009].

ZEPEDA SANCHEZ, Leopoldo: *Metodología para el Diseño Conceptual de Almacenes de Datos*, Tesis doctoral. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2008. <http://dspace.upv.es/xmlui/bitstream/handle/10251/2506/tesisUPV2841.pdf?sequence=1> [Consulta: 18-12-2009].

ANEXO 1 – GLOSARIO

Agregación de datos:	Proceso por el cual se integra un conjunto de valores de datos en un solo valor. Por lo general, la agregación se realiza durante las operaciones OLAP, pero también puede ejecutarse como parte del proceso de extracción, transformación y carga (ETC). También denominado sumarización.
Almacén de datos convencionales:	Almacén de datos que manipula sólo datos convencionales. Compárese con almacenes de datos espaciales y temporales.
Almacén de datos operacionales:	Almacén de datos centralizados que engloba toda una empresa.
Almacén de datos espaciales:	Almacén de datos que manipula datos espaciales de manera que se puedan realizar análisis espaciales. Compárese con almacén de datos convencionales y temporales.
Almacén de datos no temporal:	Almacén de datos que permite apoyo sólo de tipo temporal para las medidas. Compárese con almacén de datos temporal.
Almacén de datos temporal:	Almacén de datos que manipula información temporal (o que varía en el tiempo), de manera que se puedan realizar análisis temporales. En tanto un almacén de datos convencionales admite medidas temporales, un almacén temporal también admite niveles, jerarquías y dimensiones temporales. Compárese con almacén de datos convencionales y espaciales.
Almacén de datos:	Tipo específico de base de datos utilizado específicamente para aplicaciones analíticas. Contiene datos históricos acerca de una organización obtenidos a partir de fuentes de datos operacionales y externas.
Aplicación analítica:	Aplicación que genera información para la toma de decisiones gerenciales e incluye a menudo aspectos tales como análisis demográfico, análisis de tendencias, reconocimiento de patrones y trazado de perfiles.
Área de preparación de los datos:	Área en la cual se ejecuta el proceso de extracción, transformación y carga (ETC) y en la cual se preparan los datos origen con el propósito de incorporarlos a un almacén de datos o a un Mercado de datos.
Atributo:	Propiedad estructural de un tipo. En los modelos conceptuales, los atributos se vinculan a tipos de entidad o relación. En los modelos relacionales, los atributos definen las columnas de una relación. Un atributo contiene un nombre, una cardinalidad y un tipo de dato asociado.
Atributo compuesto:	Atributo conformado por varios otros atributos. Un atributo puede ser simple o compuesto.
Atributo convencional:	Atributo cuyo dominio consiste en un tipo de datos convencionales. Compárese con atributos espaciales y temporales.
Atributo de un solo valor:	Atributo que tiene como máximo un valor en las instancias de un tipo. Un atributo puede ser de un solo valor o multivaluado. También conocido como atributo unívoco.
Atributo espacial:	Atributo cuyo dominio es un tipo de dato espacial. Compárese con atributo convencional y temporal.
Atributo geométrico:	En el modelo MultiDim, se refiere a un atributo predefinido que almacena el contenido espacial de instancias de un tipo en un nivel.
Atributo multivaluado:	Atributo que puede tener varios valores en las instancias de un tipo. Un atributo puede tener valores múltiples o un solo valor.
Atributo no temporal:	Atributo que no lleva control de la evolución de sus valores. Compárese con atributo temporal.
Atributo o medida derivada:	Atributo o medida cuyo valor para cada instancia de un tipo se deriva por medio de una expresión de otros valores e instancias en la base de datos.
Atributo obligatorio:	Atributo al cual debe asignarse por lo menos un valor en las instancias de un tipo. Un atributo puede ser opcional u obligatorio.
Atributo opcional:	Atributo que podría no tener valor alguno en las instancias de un tipo. Un atributo es opcional u obligatorio.

Atributo simple:	Atributo que no está conformado por otros atributos, es decir, se refiere a un atributo cuya especificación incluye de manera explícita el tipo de dato asociado. Denominado también atributo atómico. Un atributo puede ser simple o compuesto.
Atributo temporal:	Atributo que lleva control de la evolución de sus valores. Compárese con atributo no temporal.
Base de datos:	Colección compartida de datos relacionados en forma lógica, con su respectiva descripción, diseñada con el fin de satisfacer las necesidades de información de una organización y apoyar las actividades de ésta.
Base de datos multidimensional:	Base de datos que representa información con base en el modelo multidimensional.
Base de datos objeto-relacional:	Base de datos que representa información de acuerdo con el modelo objeto-relacional.
Base de datos relacional:	Base de datos que representa información con base en el modelo relacional.
Cardinalidad:	Número de elementos en una colección. En los modelos conceptuales, puede referirse tanto a una especificación que restringe la colección de valores que un atributo podría tener, o bien a una colección de las instancias de un tipo determinado de relación en el cual podría participar un tipo de entidad o un nivel.
Calidad de datos:	Grado de excelencia de los datos. La calidad de los datos se determina mediante factores que establecen si éstos son coherentes, no redundantes, completos, se ciñen a las normas de la empresa, son oportunos y fáciles de entender.
Cardinalidad del tiempo de vida:	Especificación de cardinalidad cuya implementación toma en cuenta datos durante todo el período de existencia de una base de datos.
Cardinalidad instantánea:	Especificación de cardinalidad con validez en cada instante del contenido temporal de una base de datos. También se conoce como cardinalidad fotográfica.
Carga de datos:	Proceso por el cual se incorpora información al almacén de datos. La carga se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETC).
Celda:	Punto único en un cubo, definido por un grupo de coordenadas; uno para cada dimensión del cubo.
Common Warehouse Metamodel (CWM):	Metamodelo propuesto como estándar por el Grupo de Gestión de Objetos (OMG) con el fin de facilitar el intercambio de metadatos entre herramientas, plataformas y repositorios de metadatos en ambientes distribuidos heterogéneos.
Conectividad abierta de bases de datos (ODBC):	Interface normalizada de programación de aplicaciones para acceder información contenida en un sistema de administración de bases de datos independientemente del lenguaje usado en este sistema. La ODBC es similar a la conectividad Java a bases de datos (JDBC), excepto que esta última está diseñada específicamente para programas Java.
Conectividad Java a bases de datos (JDBC):	Interface de programación para aplicaciones que permite a los programas Java acceder información en un sistema de administración de bases de datos. La JDBC es similar a la conectividad de base de datos abierta (ODBC), con excepción de que este último tiene un lenguaje propio.
Consulta ad hoc:	Búsqueda de información iniciada por un usuario, según se requiera, y sobre la cual el sistema no tiene conocimiento previo. Compárese con consulta predefinida.
Consulta predefinida:	Solicitud de información específica en un formato específico realizada regularmente. Compárese con consulta ad hoc.
Cuarta forma normal (4NF):	Forma normal que garantiza que cada dependencia funcional y multivaluada en una relación esté determinada por una llave.
Cubo:	Estructura multidimensional que abarca un conjunto de medidas en cada celda. Los cubos se utilizan para implementar el procesamiento analítico en línea (OLAP). También denominado hipercubo o cubo multidimensional.

Datos actuales:	Datos obtenidos durante el período de tiempo actual que se utilizan para las operaciones rutinarias de una organización. Compárese con datos históricos.
Datos históricos:	Datos de períodos anteriores que se utiliza en el análisis de tendencias y para fines comparativos con períodos anteriores. Compárese con datos actuales.
Desnormalización:	Proceso por medio del cual se modifica el esquema de una relación de manera que no se adecúe a una forma normal específica. Si bien las relaciones no normalizadas se utilizan con frecuencia para mejorar el acceso con base en las necesidades específicas del usuario, también producen cierto grado de redundancia de datos. Compárese con normalización.
Dependencia funcional:	Restricción entre dos conjuntos de atributos en una relación que establece que las tuplas que posean los mismos valores para el primer conjunto también tienen los mismos valores para el segundo. Las dependencias funcionales se utilizan para definir varias formas normales de relaciones.
Dependencia multivaluada:	Restricción entre dos conjuntos de atributos en una relación según la cual las tuplas que poseen los mismos valores para el primer conjunto de atributos tienen un conjunto de valores posibles para el segundo grupo de atributos independientemente de que existan otros atributos en la relación. Las dependencias multivaluadas se utilizan para definir varias formas normales de relaciones.
Diagrama de esquema:	Diagrama que ilustra un esquema de acuerdo con una notación diagramática que corresponde a un modelo de datos sobre la cual se basa el esquema.
Dimensión:	Uno o varios niveles relacionados entre sí que constituyen un punto de vista específico para analizar los hechos de una base de datos multidimensional. Las dimensiones podrían estar conformadas por jerarquías.
Dimensión conformada:	Dimensión cuya semántica, estructura y uso son convenidas por toda la empresa. Comúnmente se utilizan en varios hechos de un almacén de datos y/o en varios Mercados de datos.
Dimensión convencional:	Dimensión compuesta únicamente por jerarquías convencionales. Compárese con dimensiones espaciales y temporales.
Dimensión de hechos:	Dimensión que no tiene una tabla de dimensión asociada porque todos sus atributos se encuentran en una tabla de hechos o en otras dimensiones. Denominada también dimensión degenerada.
Dimensión espacial:	Dimensión que incluye al menos una jerarquía espacial. Compárese con dimensiones convencionales y temporales.
Dimensión no temporal:	Dimensión que incluye únicamente jerarquías no temporales. Compárese con dimensión temporal.
Dimensión que cambia lentamente:	Dimensión cuyos datos cambian a lo largo del tiempo. La palabra “lentamente” subraya el hecho de que los datos en una dimensión cambian con poca frecuencia en comparación con los datos en hechos relacionados.
Dimensión temporal:	Dimensión que incluye por lo menos una jerarquía temporal. Compárese con dimensión convencional y espacial.
Diseño basado en datos:	Enfoque para diseñar un almacén de datos basado en la información disponible en los sistemas de origen. También conocido como diseño orientado a los datos. Compárese con diseño orientado al análisis.
Diseño basado en el análisis y la fuente:	Enfoque utilizado para diseñar un almacén de datos que combine los enfoques basados en el análisis y los enfoques basados en datos.
Diseño basado en el análisis:	Enfoque utilizado para diseñar un almacén de datos basado en los requisitos de análisis de los usuarios que toman decisiones o de los procesos organizacionales. Denominado también diseño basado en requisitos o diseño orientado a los negocios. Compárese con diseño basado en datos.
Disparador:	Código procedimental almacenado en un sistema de administración de base de datos que se ejecuta automáticamente en respuesta a ciertos eventos en una relación o base de datos.
Entidad:	En el modelo entidad-relación (ER) se refiere a una instancia de un tipo de entidad.

Esquema:	Especificación, de acuerdo con un modelo de datos específico, que define cómo está estructurada la información en una base o almacén de datos, el tipo de contenido que cada elemento contiene y las regulaciones que estipulan qué valores pueden ingresarse a la base o almacén de datos y cómo estos valores pueden modificarse.
Esquema conceptual:	Esquema diseñado de manera que se aproxime lo más posible a la percepción de los datos por parte de los usuarios, sin tomar en cuenta las consideraciones de implementación.
Esquema de constelación:	Esquema relacional utilizado para representar datos multidimensionales conformados por múltiples tablas de hechos que comparten tablas de dimensiones.
Esquema de copo de nieve:	Esquema relacional para representar datos multidimensionales, conformados por una tabla única y central de hechos relacionada con jerarquías de dimensión presentes en forma normal. Cada nivel de dimensión es representado por una tabla. Compárese con esquemas en estrella y esquemas de constelación.
Esquema de estrella con copo de nieve:	Esquema relacional utilizado para representar datos multidimensionales que combina los esquemas de estrella y los esquemas de copo de nieve; es decir, está conformada tanto por dimensiones normalizadas como no normalizadas.
Esquema de estrella:	Esquema relacional utilizado para representar datos multidimensionales, conformados por una tabla única y central de hechos relacionada con dimensiones no normalizadas. Cada dimensión está representada por una tabla única. Compárese con esquema de copo de nieve y esquema de constelación.
Esquema físico:	Esquema diseñado exclusivamente para maximizar la eficiencia y el funcionamiento de una determinada plataforma de bases de datos.
Esquema lógico:	Esquema cuyo diseño toma en cuenta la funcionalidad específica del tipo de sistema de administración de base de datos utilizado, tales como relacional u objeto-relacional.
Esquema objeto-relacional:	Esquema desarrollado para un sistema de administración de base de datos objeto-relacional.
Esquema relacional:	Esquema desarrollado para un sistema de administración de base de datos relacional.
Extracción de datos:	Proceso por el cual se obtiene la información a partir de fuentes de datos operacionales y externas con el fin de preparar los datos origen para su almacenamiento. La extracción se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETC).
Extracción, transformación y carga (ETC):	Proceso que permite alimentar un almacén de datos con información de una o varias fuentes. Como su nombre lo indica, es un proceso de tres pasos en el cual la información se extrae de las fuentes de datos, se transforma y, por último, se carga al almacén. El proceso de ETC también incluye la restauración del almacén de datos, con una frecuencia específica, para fines de actualización. También conocido como extracción, transformación y transporte.
Forma normal:	Conjunto de condiciones que una relación debe satisfacer para garantizar algunas propiedades ideales, por lo general para eliminar la redundancia de datos. Las formas normales más comunes son la primera, segunda, tercera, Boyce-Codd y la cuarta.
Forma normal de Boyce-Codd (FNBC):	Forma normal mediante la cual se garantiza que cada dependencia funcional en una relación esté designada por una llave.
Fuente de datos externos:	Sistema o archivo de datos de una organización distinta a la organización en cuestión, utilizado para agregar información a un almacén de datos. Compárese con fuente de datos operacionales.
Fuente de datos operacionales:	Sistema operacional que brinda información para alimentar el almacén de datos. Compárese con fuente de datos externos.
Fuente de datos:	Sistema mediante el cual se recopilan datos con el fin de integrarlos al almacén de datos. Por lo general, dicho sistema consiste en una base de datos, una aplicación, un repositorio o un archivo.

Función de agregación:	Función que calcula un valor agregado a partir de una serie de valores. En un almacén de datos, las funciones de agregación se utilizan para agregar medidas atravesando las jerarquías en sus dimensiones.
Generación de reportes:	Proceso de extraer información de una base de datos o un almacén de datos para presentarla a los usuarios en informes con gráficos, diagramas, etc.
Granularidad:	Especificación que fracciona un dominio en grupos de elementos, en el cual cada grupo se percibe como una unidad indivisible (un gránulo) a un nivel de abstracción particular. En un modelo multidimensional, se refiere al nivel de detalle en el cual se obtiene la información contenida en dimensiones y hechos.
Grupo de Gestión de Objetos (OMG):	Consorcio internacional de la industria de la computación que desarrolla normas de integración empresarial para una amplia gama de tecnologías e industrias.
Hecho conformado:	Hecho cuya semántica, dimensiones y unidades son convenidas por la empresa. Comúnmente se utilizan en varios Mercado de datos.
Hecho:	Componente central de un modelo multidimensional que contiene las medidas a ser analizadas. Los hechos se relacionan con las dimensiones.
Herencia múltiple:	Posibilidad de que un tipo de entidad tenga más de un supertipo directo.
Herencia:	Mecanismo por el cual un subtipo en una jerarquía de generalización incorpora las propiedades de sus supertipos.
Herramienta CASE (Ingeniería de Software Asistida por Computadora):	Herramienta que apoya las actividades de desarrollo de software. Las herramientas CASE pueden utilizarse en todas las etapas del ciclo de vida de desarrollo de software, incluidos el análisis, el diseño y la implementación.
Identificador del objeto:	Valor generado por un sistema que está relacionado con cada instancia de un tipo en un sistema orientado al objeto o en un sistema objeto-relacional. A cada instancia se le asigna un identificador único dentro de una base de datos. También denominado identificador artificial.
Índice:	Mecanismo para ubicar y acceder información dentro de una base de datos. Un índice puede contener una o más columnas y puede ser usado para asegurar que los valores sean únicos.
Instancia:	Elemento de un tipo de entidad o un tipo de relación. En modelos multidimensionales, las instancias de los niveles se llaman miembros.
Integración de datos:	Proceso por medio del cual se integra la información tanto a nivel de esquema como a nivel de datos proveniente de diferentes fuentes de datos. La integración se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETC).
Integridad referencial:	Restricción de integridad del modelo relacional en la cual se especifica que los valores de uno o varios atributos de una relación (llave foránea) deben incluirse en el conjunto de valores de algunos otros atributos (comúnmente la llave primaria) de la misma relación u otra diferente.
Inteligencia de negocios:	Proceso de recopilación y análisis de la información con el fin de extraer conocimiento estratégico de datos existentes en el negocio. Algunas veces este concepto se usa como sinónimo de apoyo al proceso de toma de decisiones.
Jerarquía:	Diversos niveles relacionados entre sí en una dimensión que definen rutas de agregación para las operaciones de navegación ascendente (más general) y descendente (más detallado).
Jerarquía convencional:	Jerarquía conformada únicamente por niveles convencionales. Compárese con jerarquías espaciales y temporales.
Jerarquía de generalización:	Conjunto de relaciones de generalización entre tipos de entidad. También conocido como jerarquía “es un”.
Jerarquía espacial:	Jerarquía que incluye al menos un nivel espacial. Compárese con jerarquía convencional y temporal.
Jerarquía no temporal:	Jerarquía que incluye únicamente niveles o relaciones no temporales. Compárese con jerarquía temporal.
Jerarquía temporal:	Jerarquía que incluye al menos un nivel temporal o una relación temporal. Compárese con jerarquía no temporal.

Limpieza de datos:	Proceso por el cual se transforman los datos fuente, ya sea eliminando errores e incongruencias o convirtiendo los datos a un formato estandarizado. La limpieza se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETC).
Llave compuesta:	Llave de una relación conformada por dos o más atributos.
Llave foránea:	Uno o varios atributos en una relación que están vinculados mediante integridad referencial a otros atributos de una relación. Las llaves foráneas facilitan la navegación entre relaciones. También conocida como llave externa.
Llave primaria:	Llave privilegiada, seleccionada entre todas las llaves en una relación, que se usa para representar los vínculos de la relación con otras relaciones. En el modelo relacional, toda relación debe tener una llave primaria.
Llave sustituta:	Llave primaria artificial generada por el sistema que no se deriva de ningún dato en la base de datos. Es similar al identificador de objeto en un sistema orientado al objeto.
Llave:	En un modelo conceptual se refiere a un conjunto de atributos cuyos valores identifican de forma única una instancia de un tipo de entidad o de un nivel. En un modelo relacional se refiere a una o más columnas en una relación cuyos valores identifican de forma única una fila en dicha relación. También se le conoce como identificador definido por el usuario.
Medida:	Dato específico que debe analizarse en una aplicación analítica. Las medidas están relacionadas con las celdas de un cubo.
Medida aditiva:	Medida que puede agregarse correctamente usando la función de suma con respecto a todas sus dimensiones. Es el tipo de medida más común. Compárese con medida semiaditiva o medida no aditiva.
Medida convencional:	Medida cuyo dominio consiste en un tipo de datos convencionales. Compárese con medidas espaciales y temporales.
Medida espacial:	Medida cuyo domino es un tipo de dato espacial. Compárese con medida convencional y temporal.
Medida no aditiva:	Medida que no puede agregarse correctamente usando la función de suma con respecto a una ninguna dimensión. Compárese con medidas aditivas y semiaditivas.
Medida semiaditiva:	Medida que puede agregarse correctamente usando la función de suma con respecto a algunas, aunque no todas, sus dimensiones. Compárese con medidas aditivas y no aditivas.
Medida temporal:	Medida que lleva control de la evolución de sus valores. Las medidas temporales se asemejan a los atributos temporales de los niveles. Compárese con medidas convencionales y espaciales.
Mercado de datos:	Almacén de datos especializado dirigido específicamente a un área funcional o grupo de usuarios en una organización. La información contenida en el Mercado de datos puede provenir ya sea de un almacén de datos de una empresa o directamente de fuentes de datos. Un Mercado de datos podría definirse como un pequeño almacén de datos local.
Metadatos:	Información acerca de los datos. Se refiere a información relativa a los contenidos y usos de una base de datos o un almacén de datos.
Metamodelo:	Marco de modelado que permite representar las propiedades de un lenguaje de modelado.
Método:	Secuencia de procesos comprobados que se siguen en la planificación, definición, análisis, diseño, construcción, pruebas e implementación de un sistema.
Miembro:	En un modelo multidimensional se refiere a la instancia de un nivel.
Minería de datos:	Proceso mediante el cual se analizan grandes cantidades de datos con el objeto de identificar relaciones, tendencias, patrones y asociaciones impredecibles o desconocidas que podrían ser de valor para una organización.
Modelado:	Proceso mediante el cual se construye o modifica un modelo.

Modelo:	Representación de las características esenciales de un sistema, proceso o fenómeno cuyo fin es mejorar la capacidad para entender, predecir o controlar su comportamiento.
Modelo conceptual:	Grupo de conceptos y reglas de modelado para describir esquemas conceptuales.
Modelo de datos:	Conjunto de conceptos y normas de modelado para describir esquemas de almacén de datos o de bases de datos.
Modelo de excepción de la primera forma normal (NF2):	Modelo lógico que elimina del modelo relacional las restricciones de la primera forma normal. El modelo objeto-relacional acepta relaciones no representadas en el primera forma normal.
Modelo entidad-relación (ER):	Un modelo conceptual popular definido por Peter Chen en un documento pionero de 1976.
Modelo físico:	Conjunto de conceptos y normas de modelado para describir el esquema físico de una base de datos.
Modelo lógico:	Conjunto de conceptos y normas de modelado para describir un esquema lógico. Algunos de los modelos de datos lógicos más comunes son el modelo relacional, el modelo orientado a objetos y el modelo objeto-relacional.
Modelo MultiDim:	Modelo multidimensional conceptual específico para aplicaciones de almacenes de datos y OLAP.
Modelo multidimensional:	Modelo en el cual la información se representa conceptualmente como hechos, medidas, dimensiones y jerarquías. El modelo multidimensional se utiliza para representar los requerimientos de información de las aplicaciones analíticas.
Modelo objeto-relacional:	Modelo lógico que amplía el modelo relacional con características orientadas a objetos, tales como herencia, tipos compuestos y métodos.
Modelo orientado al objeto:	Modelo lógico en el cual una aplicación se configura como un conjunto de objetos colaboradores que intercambian mensajes entre sí. Los modelos orientados a objetos incluyen características como herencia, encapsulado, polimorfismo, tipos compuestos y métodos.
Modelo relacional:	Modelo lógico en el cual la información es representada por tablas o relaciones bidimensionales.
Navegación ascendente:	Operación que genera consultas de información resumida, escalando uno o varios niveles hacia arriba de la jerarquía. Compárese con navegación descendente.
Navegación transversal:	Operación OLAP que genera consultas de datos detallados trasladándose de datos OLAP a datos origen en una base de datos relacional.
Navegación horizontal:	Operación OLAP que genera consultas sobre datos (medidas) relacionadas trasladándose de un hecho a otro a través de dimensiones comunes. También conocida como navegación inter-dimensional.
Navegación descendente:	Operación OLAP que genera consultas de datos detallados bajando uno o varios niveles en la jerarquía. Compárese con navegación ascendente. También conocido como Detallar, Desmenuzar, Desglosar, Acceso al detalle.
Nivel:	En un modelo multidimensional, se refiere a un tipo que pertenece a una dimensión. Un nivel define un conjunto de atributos y comúnmente se relaciona con otros niveles para definir jerarquías.
Nivel convencional:	Nivel que incluye únicamente atributos convencionales. Compárese con niveles espaciales y temporales.
Nivel de unión:	En el modelo MultiDim, se refiere a un nivel en el cual se conectan dos rutas alternativas de agregación.
Nivel hoja:	Nivel en una jerarquía que no está relacionado con un nivel hijo, es decir, contiene la información más detallada. Compárese con nivel raíz. También conocido como Nodo hoja.
Nivel de separación:	En el modelo MultiDim, se refiere al nivel en el cual se originan dos rutas de agregación alternativa.
Nivel espacial:	Nivel cuyos miembros tienen una extensión espacial identificada por su ubicación. Compárese con nivel convencional y temporal.

Nivel hijo:	En dos niveles relacionados entre sí dentro de una jerarquía, es el nivel inferior que contiene datos más detallados. Compárese con nivel padre.
Nivel no temporal:	Nivel que no mantiene el tiempo de vida para sus miembros. Compárese con nivel temporal.
Nivel padre:	En dos niveles relacionados entre sí en la jerarquía, el nivel padre es el nivel superior que contiene información más general. Compárese con nivel hijo.
Nivel raíz:	Nivel en una jerarquía que no tiene nivel padre, es decir, contiene la información más general. Compárese con nivel hoja.
Nivel temporal:	Nivel que almacena el tiempo de vida de sus miembros. Compárese con nivel convencional y espacial.
Normalización:	Proceso por el cual se modifica la estructura de una base de datos relacional de manera que las relaciones satisfagan algunas formas normales. Compárese con desnormalización.
Objeto:	En el mundo real, se refiere a un fenómeno cuya existencia es independiente de otros fenómenos.
OLAP híbrido (HOLAP):	Método de almacenamiento en el cual las aplicaciones OLAP se construyen sobre una base de datos multidimensional y una relacional. En este tipo de arquitectura, los datos detallados comúnmente se almacenan en la base de datos relacional, en tanto los datos agregados se almacenan en la base de datos multidimensional. Compárese con OLAP relacional y OLAP multidimensional.
OLAP multidimensional (MOLAP):	Método de almacenamiento en el cual las aplicaciones OLAP se construyen sobre una base de datos multidimensional. Compárese con OLAP relacional y OLAP híbrido.
OLAP relacional (ROLAP):	Método de almacenamiento en el cual las aplicaciones OLAP se construyen sobre una base de datos relacional. Compárese con OLAP multidimensional y OLAP híbrido.
Particionamiento:	Técnica para mejorar el funcionamiento o seguridad de una aplicación mediante la división de datos de una relación en múltiples objetos almacenados por separado. También conocida como fragmentación.
Participación obligatoria:	Especificación que indica que todas las instancias del tipo de entidad correspondiente deben participar. Puede ser opcional u obligatoria.
Participación opcional:	Función en la cual es posible que no intervengan las instancias del tipo de entidad correspondiente. Una función puede ser opcional u obligatoria.
Población:	Conjunto de instancias de un tipo de entidad o de un tipo de relación.
Primera forma normal (1NF):	Forma normal que garantiza que todos los dominios subyacentes de una relación contengan únicamente valores atómicos. Esto se logra evitando el uso de los atributos compuestos y los atributos multivaluados.
Procesamiento analítico en línea (OLAP):	Análisis interactivo de información contenida en un almacén de datos.
Procesamiento transaccional en línea (OLTP):	Trabajo orientado a transacciones en un sistema operacional que respalda las operaciones diarias de un negocio.
Proceso frontal:	Proceso que aprovecha el contenido de un almacén de datos. Una manera de lograr esto es, por ejemplo, mediante el análisis OLAP, la generación de informes y la minería de datos. Compárese con proceso posterior. También conocido como proceso interfaz de usuario.
Proceso posterior:	Proceso por el cual se incorpora al almacén de datos información proveniente de fuentes de datos operacionales y externas. Compárese con proceso frontal.
Rebanar y separar:	Operación que permite seleccionar una porción de los datos de un hecho, de acuerdo con valores específicos en una o más dimensiones.

Relación:	Estructura bidimensional para almacenar información en una base de datos relacional. Una relación tiene un número específico de columnas pero puede contener un número indeterminado de filas. Las relaciones también se conocen como tablas. También, en el modelo entidad-relación (ER), una relación es una instancia de un tipo de relación. Su existencia está sujeta a la existencia de las entidades vinculadas.
Relación de generalización:	Relación directa definida entre un supertipo y un subtipo, la cual establece que los fenómenos descritos en el subtipo son los mismos que los descritos en el supertipo, pero a un nivel de abstracción más específico (menos genérico). Las instancias del subtipo pueden sustituirse por instancias del supertipo. También se conoce como relación "es un". También conocido como relación de especialización.
Relación convencional de hechos:	Relación de hechos que requiere una operación de unión clásica entre sus dimensiones. Compárese con relaciones de hechos espaciales y temporales.
Relación de hechos:	En el modelo MultiDim se refiere a una relación que contiene datos de hechos.
Relación espacial de hechos:	Relación de hechos que requiere una operación de unión espacial entre dos o más dimensiones espaciales. Compárese con relación de hechos convencional y temporal.
Relación temporal de hechos:	Relación de hechos que requiere una operación de unión temporal entre dos o más dimensiones temporales. Compárese con relación de hechos convencional y espacial.
Relación de sincronización:	Tipo de relación que contiene una restricción temporal asociada con el tiempo de vida de las entidades vinculadas. En el modelo MultiDim, una relación de sincronización puede ser cualquiera de las siguientes: une, superpone, intersecta, contiene, adentro, cubre, cubierta por, equivale, desune, comienza, termina, precede o sigue.
Relación no temporal:	Relación que no lleva control de la evolución de las asociaciones entre sus instancias. Compárese con relación temporal.
Relación padre-hijo:	En un modelo multidimensional se refiere a un tipo de relación binaria que asocia un nivel padre con un nivel hijo dentro de una jerarquía.
Relación temporal:	Relación que lleva control de la evolución de los vínculos entre sus instancias. Compárese con relación no temporal.
Relación topológica:	Relación que tiene una restricción espacial asociada con la geometría de las entidades vinculadas. En el modelo MultiDim, una relación topológica puede ser cualquiera de las siguientes: une, superpone, intersecta, contiene, adentro, cubre, cubierta por, equivale, desune o cruza.
Actualización de datos:	Proceso por el cual se propagan las actualizaciones de las fuentes de datos al almacén de datos, con una frecuencia específica, a fin de suministrar información actualizada para el proceso de toma de decisiones. La actualización se lleva a cabo comúnmente durante el proceso de extracción, transformación y carga (ETC). También conocido como actualización, refrescamiento y restauración de datos.
Restricción de integridad:	Condición que restringe los posibles estados de una base de datos con el propósito de garantizar su consistencia de acuerdo con las normas de las aplicaciones que utilizan la base de datos.
Restricción espacial:	Restricción de integridad que impone una restricción en un ámbito espacial.
Restricción temporal:	Restricción de integridad que impone una restricción en un ámbito temporal.
Rol:	Participación de un tipo de entidad en un tipo de relación. Los nombres de los roles son obligatorios cuando el mismo tipo de entidad está relacionado más de una vez con un tipo de relación.
Segunda forma normal (2NF):	Forma normal que garantiza que una relación no contenga dependencias de llave parciales. Esto ocurre cuando una relación tiene una llave compuesta y un atributo depende de la llave sólo parcialmente.

Sistema de administración de base de datos (DBMS):	Sistema informático que permite a los usuarios definir, crear, manipular y administrar una base de datos.
Sistema de administración de base de datos relacional (RDBMS):	Sistema de administración de bases de datos en el cual la información está organizada en función del modelo relacional.
Sistema de apoyo a la toma de decisiones:	Sistema utilizado principalmente para ayudar a una organización a tomar decisiones. Respalda las operaciones analíticas sobre datos históricos. Compárese con sistema operacional. También conocido como Sistema de soporte de decisiones.
Sistema de información geográfica (SIG):	Sistema de software que permite a los usuarios definir, crear, mantener y controlar el acceso a una base de datos geográfica.
Sistema de legado:	Sistema que aún existe, que ha estado en funcionamiento por varios años y que utiliza lenguajes, plataformas y técnicas previas a la tecnología actual. Los sistemas de legado son difíciles de modificar y mantener.
Sistema gestor de bases de datos objeto/relacionales (ORDBMS):	Sistema de administración de base de datos en la cual se organiza la información en función del modelo objeto-relacional.
Sistema operacional:	Sistema que apoya las operaciones diarias de una organización. Compárese con sistema de apoyo a la toma de decisiones.
Subtipo:	En una relación de generalización se refiere al tipo más específico. Compárese con el supertipo de la relación.
Sumarización:	Característica referida a la posibilidad de agregar medidas correctamente en un nivel de jerarquía superior mediante el uso de agregaciones existentes en un nivel de jerarquía inferior.
Supertipo:	En una relación de generalización se refiere al tipo más genérico. Compárese con el subtipo de la relación.
Tabla de dimensiones:	Tabla relacional que contiene datos de dimensiones.
Tabla de hechos:	Relación que contiene datos de hechos. Una tabla de hechos por lo general tiene dos tipos de columnas: aquellas que contienen medidas y aquellas que son llaves foráneas a las tablas de dimensiones. La llave primaria de una tabla de hechos es a menudo una llave compuesta por todas sus llaves foráneas.
Tabla puente:	Tabla con una llave compuesta que representa una relación muchos a muchos, ya sea entre una tabla de hechos y una tabla de dimensión o bien entre dos tablas de dimensión que representan distintos niveles de jerarquía.
Tercera forma normal (3NF):	Forma normal que asegura que cada atributo en una relación que no es parte de una llave dependa únicamente de la llave principal y no de algún otro campo en la relación.
Tiempo bitemporal:	Especificación temporal que asocia el tiempo de validez y el tiempo de transacción con un componente particular de esquema.
Tiempo de carga (TC):	Especificación temporal que almacena datos con respecto al momento en que un elemento de información fue guardado en un almacén de datos.
Tiempo de transacción (TT):	Especificación temporal que contiene información acerca del momento en el que se almacena y elimina información de una base de datos. Compárese con tiempo válido.
Tiempo de vida:	Período de tiempo vinculado con la existencia de una instancia dentro de su tipo.
Tiempo válido (TV):	Especificación temporal que contiene información con respecto a si un elemento de datos almacenado en una base de datos es válido en la realidad percibida según los criterios de la aplicación. Compárese con tiempo de transacción.
Tipo de datos convencionales:	Tipo de datos que permite representar información alfanumérica convencional. Entre los tipos de datos convencionales más comunes se encuentran los siguientes: booleanos, cadena de caracteres, de números enteros y flotantes. Compárese con tipos de datos espaciales y temporales.

Tipo de dato espacial:	Tipo de dato que permite representar características geométricas de los fenómenos. El modelo MultiDim brinda los siguientes tipos de dato espacial: Geo, SimpleGeo, Point, Line, OrientedLine, Surface, SimpleSurface, ComplexGeo, PointSet, LineSet, OrientedLineSet, SurfaceSet y SimpleSurfaceSet. Compárese con tipos de dato convencionales y temporales.
Tipo de dato temporal:	Tipo de dato que permite representar características temporales de los fenómenos. El modelo MultiDim brinda los siguientes tipos de datos temporales: Time, SimpleTime, Instant, Interval, ComplexTime, InstantSet e IntervalSet. Compárese con tipos de dato convencionales y espaciales.
Tipo de dato:	Dominio de valores con operadores conexos. Entre los tipos de datos se incluyen los convencionales, los espaciales y los temporales.
Tipo de entidad:	En el modelo entidad-relación (ER), se refiere a la descripción de un conjunto de entidades que comparten los mismos atributos, relaciones y semántica.
Tipo de relación:	Descripción de un conjunto de relaciones que comparten los mismos atributos, roles y semántica. En el modelo MultiDim, un tipo de relación puede ser una relación de hechos o una relación padre-hijo.
Tipo de relación "n-ario":	Tipo de relación entre tres o más tipos de entidad.
Tipo de relación binaria:	Tipo de relación entre dos tipos de entidad o entre dos niveles.
Tipo de relación recursiva:	Tipo de relación en la cual el mismo tipo de entidad está asociado por dos (o más) roles. También conocida como relación recursiva.
Transformación de datos:	Manipulación de datos que permiten el cumplimiento de las normas del negocio, las regulaciones del dominio, los requisitos de integridad y otra información en el contexto del almacén de datos. La transformación de datos incluye comúnmente limpieza, agregación e integración de datos provenientes de diversas fuentes de información.
Valor nulo:	Un marcador específico que indica que el valor de un atributo está ausente, es desconocido o inaplicable.
Vista materializada:	Vista que se almacena físicamente en una base de datos. Con las vistas materializadas se logran mejores consultas puesto que permiten calcular operaciones costosas por anticipado.
Vista:	En el modelo relacional se refiere una relación virtual que se deriva de una o varias relaciones u otras vistas.

ANEXO 2 – GLOSSARY

ad hoc query: A request for information that is created by a user as the need arises and about which the system has no prior knowledge. This is to be contrasted with a predefined query.

additive measure: A measure that can be meaningfully aggregated by addition along all of its dimensions. It is the most common type of measure. This is to be contrasted with semiadditive and nonadditive measures.

aggregation function: A function that computes an aggregated value from a set of values. In a data warehouse, aggregation functions are used for aggregating measures across dimensions and hierarchies.

analysis-driven design: An approach to designing a data warehouse based on the analysis requirements of the decision-making users or organizational processes. It is also called requirements- or business-driven design. This is to be contrasted with source-driven design.

analysis/source-driven design: An approach to designing a data warehouse that is a combination of the analysis-driven and source-driven approaches.

analytical application: An application that produces information for management decisions, usually involving issues such as demographic analysis, trend analysis, pattern recognition, and profiling.

attribute: A structural property of a type. In conceptual models, attributes are attached to entity or relationship types. In the relational model, attributes define the columns of a relation. An attribute has a name, a cardinality, and an associated data type.

back-end process: A process that populates a data warehouse with data from operational and external data sources. This is to be contrasted with the front-end process.

binary relationship type: A relationship type between two entity types or between two levels.

bitemporal time: A temporal specification that associates both valid time and transaction time with a particular schema element.

Boyce-Codd normal form (BCNF): A normal form that ensures that every functional dependency in a relation is implied by a key.

bridge table: A table with a composite key that represents a many-to-many relationship either between a fact table and a dimension table or between two dimension tables representing different hierarchy levels.

business intelligence: The process of collecting and analyzing information to derive strategic knowledge from business data. Sometimes used synonymously with decision support.

cardinality: The number of elements in a collection. In conceptual models, it is either a specification that constrains the collection of values that an attribute may take, or the collection of the instances of a given relationship type in which an entity type or a level may participate.

cell: A single point in a cube defined by a set of coordinates, one for each of the cube's dimensions.

child level: Given two related levels in a hierarchy, the lower level, containing more detailed data. This is to be contrasted with the parent level.

Common Warehouse Metamodel (CWM): A metamodel proposed as a standard by the OMG to enable interchange of warehouse metadata between tools, platforms, and metadata repositories in distributed heterogeneous environments.

complex attribute: An attribute that is composed of several other attributes. An attribute is either simple or complex.

composite key: A key of a relation that is composed of two or more attributes.

computer-aided software engineering (CASE) tool: A tool that supports the activities of software development. CASE tools may be used in all phases of the software development life cycle, including analysis, design, and implementation.

conceptual model: A set of modeling concepts and rules for describing conceptual schemas.

conceptual schema: A schema that is designed to be as close as possible to the users' perception of the data, not taking into account implementation considerations.

conformed dimension: A dimension whose semantics, structure, and use are agreed upon across an enterprise. Conformed dimensions are typically used in several facts of a data warehouse and/or in several data marts.

conformed fact: A fact whose semantics, dimensions, and units are agreed upon across an enterprise. Conformed facts are typically used in several data marts.

constellation schema: A relational schema for representing multidimensional data composed of multiple fact tables that share dimension tables.

conventional attribute: An attribute that has a conventional data type as its domain. This is to be contrasted with spatial and temporal attributes.

conventional data type: A data type that allows conventional alphanumeric information to be represented. Typical conventional data types include the Boolean, string, integer, and float types. This is to be contrasted with spatial and temporal data types.

conventional data warehouse: A data warehouse that manipulates only conventional data. This is to be contrasted with spatial and temporal data warehouses.

conventional dimension: A dimension composed of only conventional hierarchies. This is to be contrasted with spatial and temporal dimensions.

conventional fact relationship: A fact relationship that requires a classical join between its dimensions. This is to be contrasted with spatial and temporal fact relationships.

conventional hierarchy: A hierarchy composed of only conventional levels. This is to be contrasted with spatial and temporal hierarchies.

conventional level: A level that includes only conventional attributes. This is to be contrasted with spatial and temporal levels.

conventional measure: A measure that has a conventional data type as its domain. This is to be contrasted with spatial and temporal measures.

cube: A multidimensional structure that contains a set of measures at each cell. Cubes are used to implement online analytical processing (OLAP). Also called a hypercube or multidimensional cube.

current data: Data from the current time period that it is used for the daily operations of an organization. This is to be contrasted with historical data.

data aggregation: The process by which a set of data values are combined into a single value. Aggregation is typically done during OLAP operations but is also done as part of the ETL process. It is also called summarization.

data cleaning: The process of transforming source data by removing errors

and inconsistencies or by converting it into a standardized format, typically done as part of the ETL process.

data extraction: The process of obtaining data from operational and external data sources in order to prepare the source data for a data warehouse. Extraction is typically done as part of the ETL process.

data integration: The process of reconciling data coming from different data sources, at both the schema and the data level. Integration is typically done as part of the ETL process.

data loading: The process of populating a data warehouse, typically done as part of the ETL process.

data mart: A specialized data warehouse that is targeted at a particular functional area or user group in an organization. The data in a data mart

can either be derived from an enterprise-wide data warehouse or be collected directly from data sources. A data mart can be seen as a small, local data warehouse.

data mining: The process of analyzing large amounts of data to identify unsuspected or unknown relationships, trends, patterns, and associations that might be of value to an organization.

data model: A set of modeling concepts and rules for describing database or data warehouse schemas.

data quality: The degree of excellence of data. Various factors contribute to data quality, such as whether the data is consistent, is nonredundant, is complete, follows business rules, is timely, and is well understood.

data refreshing: The process of propagating updates from the data sources to the data warehouse at a specified frequency in order to provide up-todate data for the decision-making process. Refreshing is typically done as part of the ETL process.

data source: A system from which data is collected in order to be integrated into a data warehouse. Typically, such a system could be a database, an application, a repository, or a file.

data staging area: An area where the ETL process is executed and where the source data is prepared in order to be introduced into a data warehouse or data mart.

data transformation: The manipulation of data to bring it into conformance with business rules, domain rules, integrity rules, and other data within the warehouse environment. Data transformation typically includes cleaning, aggregating, and integrating data from several data sources.

data type: A domain of values with associated operators. The data types in this book include conventional data types, spatial data types, and temporal data types.

data warehouse: A specific type of database that is targeted at analytical applications. It contains historical data about an organization obtained from operational and external data sources.

database: A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization and to support its activities.

database management system (DBMS): A software system that allows users to define, create, manipulate, and manage a database.

decision support system: A system that is primarily used to assist an organization in making decisions. It supports analytical operations on historical data. This is to be contrasted with an operational system.

denormalization: The process of modifying the schema of a relation so that it does not conform to a particular normal form. Denormalized relations are often used to improve access for specific user needs, but they result in some degree of data redundancy. This is to be contrasted with normalization.

derived attribute or measure: An attribute or measure whose value for each instance of a type is derived, by means of an expression, from other values and instances in the database.

dimension: One or several related levels that constitute a specific viewpoint for analyzing the facts of a multidimensional database. Dimensions may be composed of hierarchies.

dimension table: A relational table that contains dimension data.

drill-across: An OLAP operation that queries related data, moving from one fact to another through common dimensions.

drill-down: An OLAP operation that queries detailed data, moving down one or several hierarchy levels. This is to be contrasted with roll-up.

drill-through: An OLAP operation that queries detailed data, moving from OLAP data to the source data in a relational database.

enterprise data warehouse: A centralized data warehouse that encompasses an entire enterprise.

entity: In the ER model, an instance of an entity type.

entity-relationship (ER) model: A popular conceptual model defined by Peter Chen in a seminal paper of 1976.

entity type: In the ER model, a description of a set of entities that share the same attributes, relationships, and semantics.

external data source: A system or data file from an organization external to the organization under consideration that is used for providing data to a data warehouse. This is to be contrasted with an operational data source.

extraction-transformation-loading (ETL): The process that allows a data warehouse to be populated from one or several data sources. As the name indicates, it is a three-step process of extracting data from the data sources, transforming the data, and finally loading the data into a data warehouse. The ETL process also includes refreshing the data warehouse at a specified frequency in order to make it up to date.

fact: A central component of a multidimensional model that contains the measures to be analyzed. Facts are related to dimensions.

fact dimension: A dimension that does not have an associated dimension table because all its attributes are in a fact table or in other dimensions. It is also called a degenerate dimension.

fact relationship: In the MultiDim model, a relationship that contains fact data.

fact table: A relation that contains fact data. A fact table typically has two types of columns: those that contain measures and those that are foreign keys to dimension tables. The primary key of a fact table is usually a composite key that is made up of all of its foreign keys.

first normal form (1NF): A normal form that ensures that all underlying domains of a relation contain only atomic values. This is achieved by disallowing complex and multivalued attributes.

foreign key: One or several attributes in a relation that are related through referential integrity to other attributes of a relation. Foreign keys support navigation between relations.

fourth normal form (4NF): A normal form that ensures that every functional and multivalued dependency in a relation is implied by a key.

front-end process: A process that exploits the contents of a data warehouse. This can be done in many ways, including OLAP analysis, reporting, and data mining. This is to be contrasted with a back-end process.

functional dependency: A constraint between two sets of attributes in a relation stating that the tuples that have the same values for the first set of attributes also have the same values for the second set of attributes. Functional dependencies are used to define various normal forms of relations.

generalization hierarchy: A set of generalization relationships between entity types. It is also called an is-a hierarchy.

generalization relationship: A directed relationship, defined between a supertype and a subtype, stating that phenomena described in the subtype are the same as those described in the supertype, but at a more specific (less generic) abstraction level. Instances of the subtype are substitutable for instances of the supertype. It is also called an is-a relationship.

geographic information system (GIS): A software system that allows users to define, create, maintain, and control access to a geographic database.

geometry attribute: In the MultiDim model, a predefined attribute that stores the spatial extent of instances of a type or a level.

granularity: A specification that partitions a domain into groups of elements, where each group is perceived as an indivisible unit (a granule) at a particular abstraction level. In a multidimensional model, it is the level of detail at which data is captured in dimensions and facts.

hierarchy: Several related levels of a dimension that define aggregation paths for roll-up and drill-down operations.

historical data: Data from previous time periods that is used for trend analysis and for comparison with previous periods. This is to be contrasted with current data.

hybrid OLAP (HOLAP): A storage method in which OLAP applications are built on top of both a relational and a multidimensional database. In such an architecture, the detailed data is typically stored in the relational database, while the aggregated data is stored in the multidimensional database. This is to be contrasted with relational OLAP and multidimensional OLAP.

index: A mechanism to locate and access data within a database. An index may involve one or more columns and be a means of enforcing uniqueness of their values.

inheritance: The mechanism by which a subtype in a generalization hierarchy incorporates the properties of its supertypes.

instance: An element of an entity or a relationship type. In multidimensional models, instances of levels are members.

instant cardinality: A cardinality specification that is valid at each instant of the temporal extent of a database. It is also called the snapshot cardinality.

integrity constraint: A condition that restricts the possible states of a database in order to enforce their consistency with the rules of the applications using the database.

Java Database Connectivity (JDBC): An application programming interface that enables Java programs to access data in a database man agement system. JDBC is similar to ODBC, but ODBC is languageindependent.

joining level: In the MultiDim model, a level in which two alternative aggregation paths are merged.

key: In a conceptual model, a set of attributes whose values uniquely identify an instance of an entity type or a level. In the relational model, a set of one or more columns in a relation whose values uniquely identify a row in that relation. It is also called a user-defined identifier.

leaf level: A level in a hierarchy that is not related to a child level, i.e., it contains the most detailed data. This is to be contrasted with a root level.

legacy system: An existing system that has been in place for several years and uses languages, platforms, and techniques prior to current technology. Legacy systems are difficult to modify and maintain.

level: In a multidimensional model, a type belonging to a dimension. A level defines a set of attributes and is typically related to other levels to define hierarchies.

lifespan (LS): The time frame that is associated with the membership of an instance into its type.

lifespan cardinality: A cardinality specification whose enforcement takes account of data over the whole temporal extent of a database.

loading time (LT): A temporal specification that stores information about when a data element was stored in a data warehouse.

logical model: A set of modeling concepts and rules for describing a logical schema. Some typical logical data models are the relational model, the object-oriented model, and the object-relational model.

logical schema: A schema whose design takes into account the specific functionality of the type of database management system used, such as relational or object-relational.

materialized view: A view which is physically stored in a database. Materialized views allow query performance to be enhanced by precalculating costly operations.

mandatory attribute: An attribute that must be given at least one value in the instances of a type. An attribute is either optional or mandatory.

mandatory role: A role in which instances of the corresponding entity type must participate. A role is either optional or mandatory.

measure: A particular piece of information that has to be analyzed in an analytical application. Measures are associated to cells in a cube.

member: In a multidimensional model, an instance of a level.

metadata: Literally, data about data. It is information about the contents and uses of a database or data warehouse.

metamodel: A modeling framework that allows one to represent the properties of a modeling language.

method: A sequence of proven processes followed in planning, defining, analyzing, designing, building, testing, and implementing a system.

model: A representation of the essential characteristics of a system, process, or phenomenon intended to enhance our ability to understand, predict, or control its behavior.

modeling: The process of constructing or modifying a model.

monovalued attribute: An attribute that may have at most one value in the instances of a type. An attribute is either monovalued or multivalued.

MultiDim model: A particular conceptual multidimensional model for data warehouse and OLAP applications.

multidimensional database: A database that represents data according to the multidimensional model.

multidimensional model: A model in which information is conceptually represented as facts, measures, dimensions, and hierarchies. The multidimensional model is used to represent the information requirements of analytical applications.

multidimensional OLAP (MOLAP): A storage method in which OLAP applications are built on top of a multidimensional database. This is to be contrasted with relational OLAP and hybrid OLAP.

multiple inheritance: The possibility for an entity type to have more than one direct supertype.

multivalued attribute: An attribute that may have several values in the instances of a type. An attribute is either multivalued or monovalued.

multivalued dependency: A constraint between two sets of attributes in a relation stating that the tuples that have the same values for the first set of attributes have a set of possible values for the second set of attributes independently of any other attributes of the relation. Multivalued dependencies are used to define various normal forms of relations.

n-ary relationship type: A relationship type among three or more entity types.

non-first-normal-form (NF2) model: A logical model that removes from the relational model the restrictions of the first normal form. The objectrelational model allows non-first-normal-form relations.

nonadditive measure: A measure that cannot be meaningfully aggregated by addition across any dimension. This is to be contrasted with additive and semiadditive measures.

nontemporal attribute: An attribute that does not keep track of the evolution of its values. This is to be contrasted with a temporal attribute.

nontemporal data warehouse: A data warehouse that allows only temporal support for measures. This is to be contrasted with a temporal data warehouse.

nontemporal dimension: A dimension that includes only nontemporal hierarchies. This is to be contrasted with a temporal dimension.

nontemporal hierarchy: A hierarchy that includes only nontemporal levels or relationships. This is to be contrasted with a temporal hierarchy.

nontemporal level: A level that does not keep the lifespan for its members. This is to be contrasted with a temporal level.

nontemporal relationship: A relationship that does not keep track of the evolution of the links between its instances. This is to be contrasted with a temporal relationship.

normal form: A set of conditions that a relation must satisfy to guarantee some desirable properties, typically to eliminate data redundancy. The most usual normal forms are the first, second, third, Boyce-Codd, and fourth normal forms.

normalization: The process of modifying the structure of a relational database so that the relations satisfy some normal forms. This is to be contrasted with denormalization.

null value: A particular marker that indicates that the value of an attribute is missing, unknown, or inapplicable.

object: In the real world, a phenomenon that is perceived as having some existence independently of other phenomena.

object identifier (oid): A system-generated value that is associated with each instance of a type in an object-oriented or an object-relational system. Each instance is given an identifier that is unique within a database. It is also called a surrogate.

Object Management Group (OMG): An international computer industry consortium that is developing enterprise integration standards for a wide range of technologies and industries.

object-oriented model: A logical model in which an application is modeled as a set of cooperating objects that exchange messages between them. Object-oriented models include features such as inheritance, encapsulation, polymorphism, complex types, and methods.

object-relational database: A database that represents data according to the object-relational model.

object-relational database management system (ORDBMS): A database management system in which data is organized according to the object-relational model.

object-relational model: A logical model that extends the relational model with object-oriented features, such as inheritance, complex types, and methods.

object-relational schema: A schema targeted at an object-relational database management system.

online analytical processing (OLAP): Interactive analysis of data contained in a data warehouse.

online transaction processing (OLTP): Transaction-oriented work on an operational system that supports daily business operations.

Open Database Connectivity (ODBC): A standardized application programming interface for accessing data in a database management system in a language-independent manner. ODBC is similar to JDBC, but JDBC is designed specifically for Java programs.

operational data source: An operational system that provides data for a data warehouse. This is to be contrasted with an external data source.

operational system: A system that supports the daily operations of an organization. This is to be contrasted with a decision support system.

optional attribute: An attribute that may have no value in the instances of a type. An attribute is either optional or mandatory.

optional role: A role in which instances of the corresponding entity type may not participate. A role is either optional or mandatory.

parent-child relationship: In a multidimensional model, a binary relationship type that links a parent and a child level in a hierarchy.

parent level: Given two related levels in a hierarchy, the upper level, containing more general data. This is to be contrasted with the child level.

partitioning: A technique to improve the performance or security of an application by splitting data in a relation into multiple objects that are stored separately. It is also called fragmentation.

physical model: A set of modeling concepts and rules for describing the physical schema of a database.

physical schema: A schema customized to maximize efficiency and performance on a particular database platform.

population: The set of instances of an entity or of a relationship type.

predefined query: A request for specific information in a specific format that is performed on a regular basis. This is to be contrasted with an ad hoc query.

primary key: A privileged key, selected from all the keys in a relation, that is used to represent the links of the relation to other relations. In the relational model, every relation must have a primary key.

recursive relationship type: A relationship type in which the same entity type is linked by two (or more) roles.

referential integrity: An integrity constraint of the relational model specifying that the values of one or several attributes of a relation (the foreign key) must be included in the set of values of some other attributes (typically the primary key) of the same or another relation.

relation: A two-dimensional structure for storing information in a relational database. A relation has a specified number of columns but can have any number of rows. Relations are also called tables.

relational database: A database that represents data according to the relational model.

relational database management system (RDBMS): A database management system in which data is organized according to the relational model.

relational model: A logical model in which information is represented using two-dimensional tables or relations.

relational OLAP (ROLAP): A storage method in which OLAP applications are built on top of a relational database. This is to be contrasted with multidimensional OLAP and hybrid OLAP.

relational schema: A schema targeted at a relational database management system.

relationship: In the ER model, an instance of a relationship type. Its existence is subject to the existence of the linked entities.

relationship type: A description of a set of relationships that share the same attributes, roles, and semantics. In the MultiDim model, a relationship type can be either a fact relationship or a parent-child relationship.

reporting: The process of extracting data from a database or a data warehouse and presenting it to users in reports containing graphs, charts, etc.

role: The participation of an entity type in a relationship type. Role names are mandatory when the same entity type is related more than once to a relationship type.

roll-up: An operation that queries summarized data, moving up one or several hierarchy levels. This is to be contrasted with drill-down.

root level: A level in a hierarchy that does not have a parent level, i.e., it contains the most general data. This is to be contrasted with a leaf level.

schema: A specification, according to a given data model, that includes a definition of how the data in a database or data warehouse is structured, the type of content that each data element can contain, and the rules that govern what data values may be entered in the database or data warehouse and how these values can evolve.

schema diagram: A diagram that illustrates a schema according to a diagrammatic notation that corresponds to the data model on which the schema is based.

second normal form (2NF): A normal form that ensures that a relation contains no partial key dependencies. This is the case when a relation has a composite key and an attribute is dependent upon only part of the key.

semiadditive measure: A measure that can be meaningfully aggregated by addition along some, but not all, of its dimensions. This is to be contrasted with additive and nonadditive measures.

simple attribute: An attribute that is not composed of other attributes, i.e., an attribute whose specification explicitly includes the associated data type. It is also called an atomic attribute. An attribute is either simple or complex.

slice-and-dice: An operation that allows one to select a portion of the data of a fact on the basis of specified values in one or several dimensions.

slowly changing dimension: A dimension whose data changes over time. The term slowly emphasizes the fact that the data in a dimension changes less frequently than the data in related facts.

snowflake schema: A relational schema for representing multidimensional data, composed of a single, central fact table related by normalized dimension hierarchies. Each dimension level is represented in a table. This is to be contrasted with star and starflake schemas.

source-driven design: An approach to designing a data warehouse based on the data available in the underlying source systems. It is also called

data-driven design. This is to be contrasted with analysis-driven design.

spatial attribute: An attribute that has a spatial data type as its domain. This is to be contrasted with conventional and temporal attributes.

spatial constraint: An integrity constraint that imposes a restriction on a spatial extent.

spatial data type: A data type that allows one to represent geometric features of phenomena. The MultiDim model provides the following spatial data types: Geo, SimpleGeo, Point, Line, OrientedLine, Surface, SimpleSurface, ComplexGeo, PointSet, LineSet, OrientedLineSet, SurfaceSet, and SimpleSurfaceSet. This is to be contrasted with conventional and temporal data types.

spatial data warehouse: A data warehouse that manipulates spatial data, thus allowing spatial analysis. This is to be contrasted with conventional and temporal data warehouses.

spatial dimension: A dimension that includes at least one spatial hierarchy. This is to be contrasted with conventional and temporal dimensions.

spatial fact relationship: A fact relationship that requires a spatial join between two or more spatial dimensions. This is to be contrasted with conventional and temporal fact relationships.

spatial hierarchy: A hierarchy that includes at least one spatial level. This is to be contrasted with conventional and temporal hierarchies.

spatial level: A level whose members have a spatial extent that keeps track of their location. This is to be contrasted with conventional and temporal levels.

spatial measure: A measure that has a spatial data type as its domain. This is to be contrasted with conventional and temporal measures.

splitting level: In the MultiDim model, a level in which two alternative aggregation paths start.

star schema: A relational schema for representing multidimensional data, composed of a single, central fact table related to denormalized dimensions.

Each dimension is represented in a single table. This is to be contrasted with snowflake and starflake schemas.

starflake schema: A relational schema for representing multidimensional data that is a combination of the star and snowflake schemas, i.e., it is composed of both normalized and denormalized dimensions.

subtype: In a generalization relationship, the most specific type. This is to be contrasted with the supertype of the relationship.

summarizability: A characteristic referring to the possibility of correctly aggregating measures in a higher hierarchy level by taking into account existing aggregations in a lower hierarchy level.

supertype: In a generalization relationship, the most generic type. This is to be contrasted with the subtype of the relationship.

surrogate key: A system-generated artificial primary key that is not derived from any data in the database. It is similar to the object identifier in an object-oriented system.

synchronization relationship: A relationship type that has an associated temporal constraint on the lifespans of the linked entities. In the Multi-Dim model, a synchronization relationship may be one of the following: meets, overlaps, intersects, contains, inside, covers, coveredBy, equals, disjoint, starts, finishes, precedes, or succeeds.

temporal attribute: An attribute that keeps track of the evolution of its values. This is to be contrasted with a nontemporal attribute.

temporal constraint: An integrity constraint that imposes a restriction on a temporal extent.

temporal data type: A data type that allows one to represent temporal features of phenomena. The MultiDim model provides the following temporal

data types: Time, SimpleTime, Instant, Interval, ComplexTime, InstantSet, and IntervalSet. This is to be contrasted with conventional and spatial data types.

temporal data warehouse: A data warehouse that manages temporal (or time-varying) data, thus allowing temporal analysis. While a conventional data warehouse supports temporal measures, a temporal data warehouse also supports temporal levels, hierarchies, and dimensions. This is to be contrasted with conventional and spatial data warehouses.

temporal dimension: A dimension that includes at least one temporal hierarchy. This is to be contrasted with conventional and spatial dimensions.

temporal fact relationship: A fact relationship that requires a temporal join between two or more temporal dimensions. This is to be contrasted with conventional and spatial fact relationships.

temporal hierarchy: A hierarchy that includes at least one temporal level or one temporal relationship. This is to be contrasted with a nontemporal hierarchy.

temporal level: A level that stores the lifespans of its members. This is to be contrasted with conventional and spatial levels.

temporal measure: A measure that keeps track of the evolution of its values. Temporal measures are similar to temporal attributes for levels. This is to be contrasted with conventional and spatial measures.

temporal relationship: A relationship that keeps track of the evolution of the links between its instances. This is to be contrasted with a nontemporal relationship.

third normal form (3NF): A normal form that ensures that every attribute in a relation that it is not part of a key is dependent only on the primary key, and not on any other field in the relation.

topological relationship: A relationship that has an associated spatial constraint on the geometries of the linked entities. In the MultiDim model, a topological relationship may be one of the following: meets, overlaps, intersects, contains, inside, covers, coveredBy, equals, disjoint, or crosses.

transaction time (TT): A temporal specification that keeps information about when a data element is stored in and deleted from a database. This is to be contrasted with valid time.

trigger: A procedural code stored in a database management system that is automatically executed in response to certain events in a relation or a database.

valid time (VT): A temporal specification that keeps information about when a data element stored in a database is considered valid in the reality perceived from the point of view of the application. This is to be contrasted with transaction time.

view: In the relational model, a virtual relation that is derived from one or several relations or other views.

ANEXO 3 –NUEVOS TÉRMINOS PROPUESTOS

1. Conteo doble	Problema que ocurre cuando una agregación de hechos en un conjunto de valores de dimensión se solapa debido a una relación muchos a muchos.
2. Dimensión de valores múltiples	Se refiere a la situación en la cual varios miembros de una dimensión participan en la misma instancia de una relación de hechos.
3. Dimensión degenerada	Llave de dimensión que no tiene atributos y, por lo tanto, no se vincula con una tabla de dimensiones. Por lo general, juegan un rol integral en la llave primaria de una tabla de hechos. También se conoce como dimensión de hechos.
4. Dimensión multivaluada	Una tabla de hechos posee sólo conexiones a dimensiones que representan un valor único, tal como un producto o tiempo único, pero algunas veces es posible conectar un registro de la tabla de hechos a una dimensión que represente un número de valores abiertos como, por ejemplo, el número de diagnósticos simultáneos que un paciente podría tener al momento de recibir un tratamiento. En este caso estamos hablando de que la tabla de hechos tiene una dimensión multivaluada.
5. Dimensión que juega diferentes roles	Dimensión que a menudo se recicla para aplicaciones múltiples dentro de la misma base de datos. Por ejemplo, la dimensión "Fecha" puede utilizarse para "Fecha de venta", "Fecha de entrega" o "Fecha de contrato".
6. Jerarquía alternativa	Jerarquía conectada a una dimensión con niveles compartidos.
7. Jerarquía balanceada	Jerarquía con niveles y ramas significativos que poseen una profundidad. El padre lógico de cada nivel se encuentra en el nivel directamente arriba de él. En Oracle OLAP también se les conoce como jerarquías normales.
8. Jerarquía desigual	Jerarquía en la cual cada nivel tiene un significado coherente, pero cuyas ramas tienen profundidades incoherentes debido a que por lo menos un atributo miembro en un nivel rama no ha sido introducido. En Analysis Services y Oracle OLAP también se le conoce como jerarquía no cubierta.
9. Jerarquía disjunta	Tipo de jerarquía que no comparte niveles. Forman parte de las jerarquías paralelas independientes.
10. Jerarquía estricta	También conocida como jerarquía asimétrica, la jerarquía estricta es aquella en la que cada miembro hijo posee un único miembro padre.
11. Jeraquía no balanceada	Jerarquía en la cual existen diferentes niveles entre el nivel superior y los miembros hoja. Una jerarquía no balanceada puede utilizarse para representar, por ejemplo, un organigrama. En Oracle OLAP también se les conoce como jerarquías desiguales.
12. Jerarquía no cubierta	Instancia que no posee un nivel intermedio.
13. Jerarquía no estricta	Jerarquía en la cual existe una relación muchos a muchos entre el nivel padre y el nivel hijo dentro de una jerarquía.
14. Jerarquía non-onto	Jerarquía simple y asimétrica que permite manejar miembros sin hijos en categorías superiores.
15. Jerarquías definidas por el usuario o jerarquías de niveles múltiples	Nombre que el programa Analysis Services le asigna a las jerarquías. Se trata de una jerarquía balanceada de jerarquías atributo utilizada que permite a los usuarios navegar en los datos de cubo.
16. Jeraquías paralelas dependientes	Las jerarquías paralelas dependientes están compuestas por varias jerarquías que utilizan diferentes criterios de análisis y comparten algunos niveles.
17. Jeraquías paralelas no dependientes	Las jerarquías paralelas independientes están compuestas por jerarquías disjuntas, es decir, que no comparten niveles. Cada jerarquía componente puede ser de diferente tipo.

18. Jerarquías que omiten niveles	Nombre que Analysis Services y Oracle OLAP le asigna a un tipo de jerarquía no cubierta. Se trata de una jerarquía que contiene por lo menos un miembro cuyos padres se encuentra al menos un nivel arriba de él, lo cual crea un hueco en la jerarquía. Por ejemplo, en una dimensión Geografía con niveles para Ciudad, Estado y País, Washington D.C. es una ciudad que no tiene un valor Estado; su padre es Estados Unidos en el nivel País.
19. Jerarquía recursiva	Jerarquía que utiliza relaciones padre-hijo inherentes entre sus niveles. Una jerarquía no balanceada que emplea un despliegue recursivo se representa mediante pares de nivel padre-hijo.
20. Mapeo de datos	Proceso por medio del cual se asigna un elemento de los datos fuente a un elemento de datos meta.
21. Miembro no hoja	Miembro que tiene uno o más miembros secundarios.
22. Operación de mezcla	Consulta que selecciona datos de una o más tablas.
23. Tabla plana	También conocida como tabla descentralizada, este tipo de tabla incluye toda la información en una tabla única independientemente de si existe o no una relación. Esto se traduce en valores que se repiten en registros sucesivos (en otras palabras, la estructura es denormalizada). La tabla plana se utiliza principalmente para desplegar información o generar reportes.
24. Tabla tipada	Una tabla tipada es una especialización de una tabla base o de una vista, pero no ambas en forma simultánea. Las tablas tipadas permiten utilizar los identificadores generados por el sistema para enlazar filas entre tablas. Las tablas tipadas también permiten definir métodos para los tipos asociados con las tablas.

ANEXO 4 – CAPÍTULO III

3 Almacenes de datos convencionales

Las ventajas de usar modelos conceptuales para diseñar aplicaciones son bien conocidas. En particular, los modelos conceptuales facilitan la comunicación entre usuarios y diseñadores, por cuanto no es necesario tener conocimientos acerca de las características específicas que subyacen a la plataforma de implementación. Asimismo, los esquemas que se desarrollan usando modelos conceptuales pueden mapearse a varios modelos lógicos, tales como los modelos relacionales, objeto-relacionales u orientados al objeto, lo cual simplifica las respuestas a cambios en la tecnología utilizada. Además, los modelos conceptuales facilitan el mantenimiento y evolución de las aplicaciones, pues se centran en las necesidades de los usuarios. Por consiguiente, ayudan a efectuar cambios ulteriores en los esquemas lógicos y de implementación con mayor facilidad.

En este capítulo presentamos el modelo MultiDim, un modelo multidimensional conceptual que permite representar requerimientos de datos para aplicaciones de almacenes de datos y OLAP. La definición del modelo se explica en la Sección 3.1. En vista de que las jerarquías son esenciales para aprovechar al máximo los sistemas de almacenes de datos y OLAP, en la Sección 3.2 consideramos varios tipos de jerarquías que se presentan en situaciones de la vida real. Clasificamos estas jerarquías mediante una representación gráfica, haciendo hincapié en las diferencias entre algunos tipos de jerarquías que la literatura actual no aborda. En la Sección 3.3 examinamos algunos aspectos avanzados de modelado, concretamente las jerarquías complejas y las dimensiones que juegan diferentes roles, las dimensiones de hechos y las dimensiones de valores múltiples. El metamodelo de nuestro modelo conceptual multidimensional se ilustra en la Sección 3.4.

Posteriormente ofrecemos un mapeo de nuestro modelo conceptual a los modelos relacionales y objeto-relacionales, mostrando de esta manera la factibilidad de implementar nuestro modelo en sistemas actuales de gestión de bases de datos. En la Sección 3.5 empezamos por describir algunas reglas generales de mapeo para el modelo conceptual. Luego, en la Sección 3.6 incluimos el análisis de mapeos específicos de diferentes tipos de jerarquías comparando las alternativas de mapeo para algunas de ellas. En la Sección 3.7 nos referimos a la implementación de las jerarquías en plataformas comerciales utilizando como ejemplos Microsoft Analysis Services 2005 y Oracle OLAP 10g. Por último, en la Sección 3.8 nos referimos a los trabajos relacionados con jerarquías y modelos conceptuales multidimensionales y, en la Sección 3.9, resumimos este capítulo.

3.1 MultiDim: Un modelo multidimensional conceptual

En esta sección definimos el modelo MultiDim: un modelo multidimensional conceptual. Nuestro modelo nos permite representar, a nivel conceptual, todos los elementos que se requieren en las aplicaciones de almacenes de datos y OLAP, es decir, dimensiones, jerarquías y hechos con medidas asociadas. La notación gráfica del modelo MultiDim que se ilustra en la Figura 3.1[5] se asemeja a la del modelo entidad-relación (ER).

[5] En el Apéndice B se incluye una descripción más exhaustiva de la notación usada.

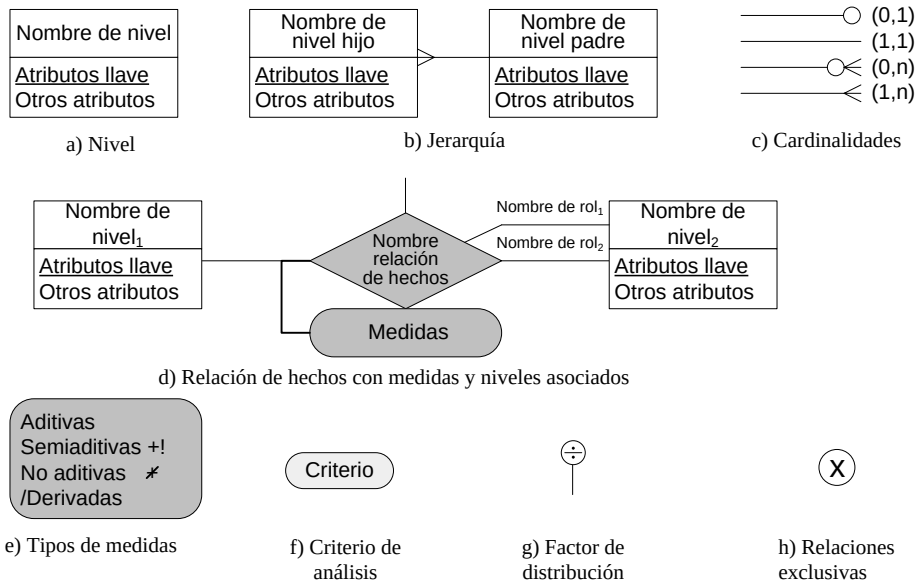


Figura 3.1. Notación del modelo MultiDim

Con el fin de brindar un panorama general del modelo, utilizaremos el ejemplo de la Figura 3.2, en el cual se ilustra el esquema conceptual de un almacén de datos para ventas. Dicha figura incluye varios tipos de jerarquías que presentaremos en forma más exhaustiva en las secciones subsiguientes.

Un *esquema* se compone de un conjunto de dimensiones y un conjunto de relaciones de hecho.

Una *dimensión* es un concepto abstracto que agrupa los datos que poseen un significado semántico común dentro del dominio a modelar. Una dimensión se compone de un conjunto de jerarquías (véase más adelante) y una jerarquía se compone, a su vez, de un conjunto de niveles.

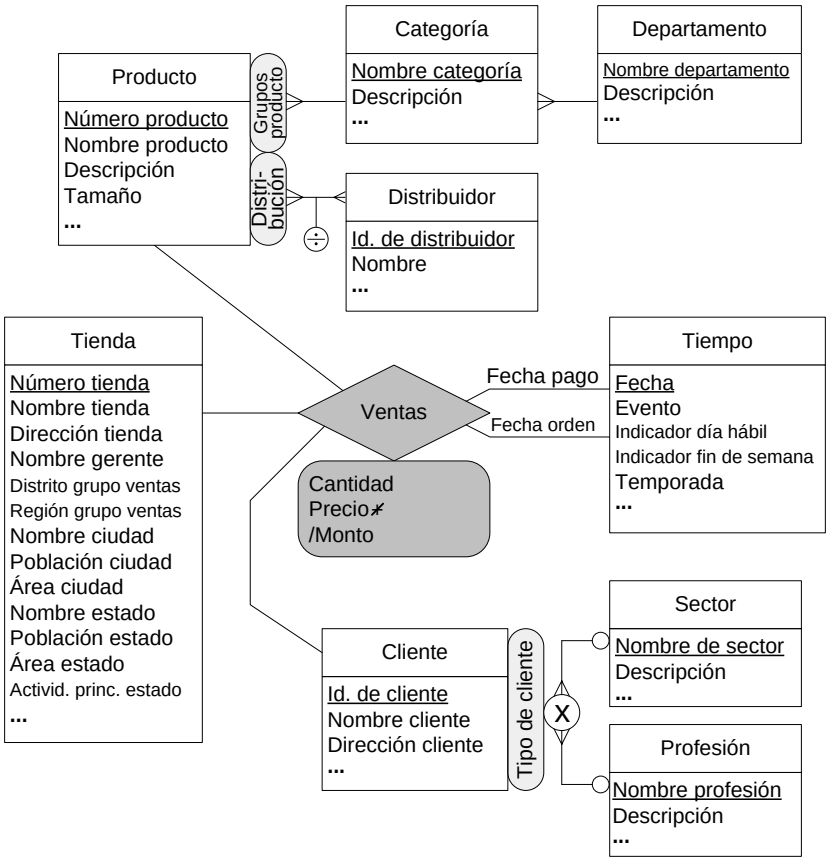


Figura 3.2. Esquema multidimensional conceptual de un almacén de datos para ventas

Un *nivel* corresponde a un tipo de entidad en el modelo ER y describe un conjunto de conceptos de la vida real que, desde la perspectiva de la aplicación, poseen características similares. Por ejemplo, Producto, Categoría y Departamento son algunos de los niveles que se ilustran en la Figura 3.2. Las instancias de un nivel se denominan *miembros*. Como se muestra en la Figura 3.1a, un nivel está compuesto por un conjunto de *atributos* que describen las características de sus miembros. Adicionalmente, un nivel tiene una o varias *llaves* que identifican de manera única a los miembros de este nivel; cada llave posee uno o más atributos. Por ejemplo, en la Figura 3.2, Nombre categoría es un atributo llave del nivel Categoría. A cada atributo de un nivel le corresponde un tipo, es decir, un dominio para sus valores. Los dominios típicos son de valores enteros, reales y de cadena de caracteres. A efectos de brevedad, la representación gráfica de nuestros esquemas conceptuales excluye información sobre tipos relativos a atributos. Si fuera necesario podría incluirse, aunque sí la incluimos en la representación textual de nuestro modelo que aparece en el Apéndice A.

Una *relación de hechos* (Figura 3.1d) expresa un enfoque de análisis y representa una relación *n*-aria entre niveles. Por ejemplo, la relación de hechos Ventas entre los niveles Producto, Tienda, Cliente y Tiempo, que se ilustra en la Figura 3.2, se utiliza para analizar los datos relacionados con ventas. Las instancias de una relación de hechos se denominan *hechos*. Dado que la cardinalidad de cada nivel que participa en una relación de hechos es (0,n), omitimos

dichas cardinalidades para simplificar el modelo. Además, según se muestra en la Figura 3.1d, el mismo nivel puede participar varias veces en una relación de hechos desempeñando *roles* diferentes. Cada rol se identifica con un nombre y se representa a través de un enlace separado entre el nivel correspondiente y la relación de hechos. Por ejemplo, en la Figura 3.2, el nivel Tiempo participa en la relación de hechos Ventas con los roles Fecha pago y Fecha orden.

Una relación de hechos podría tener atributos comúnmente denominados *medidas*, las cuales contienen datos (por lo general numéricos) que se analizan utilizando las diversas perspectivas representadas por las dimensiones. Por ejemplo, la relación de hechos Ventas, que se ilustra en la Figura 3.2, incluye las medidas Cantidad, Precio y Monto. Los atributos llave de los niveles que participan en una relación de hechos indican la granularidad de las medidas, es decir, el nivel de detalle en el cual se representan las medidas.

En la Sección 2.6.2 clasificamos las medidas como *aditivas*, *semiaditivas* o *no aditivas*. Tal como se ilustra en la Figura 3.1e, asumimos por defecto que las medidas son aditivas, es decir, pueden sumarse en todas las dimensiones. Para las medidas semiaditivas y no aditivas, agregamos los símbolos +! y ≠, respectivamente, a la par del nombre de la medida. Asimismo, las medidas y los atributos de nivel pueden ser *derivados* cuando se calculan sobre la base de otras medidas o atributos en el esquema. Utilizamos el símbolo / para indicar medidas y atributos derivados. Por ejemplo, en la Figura 3.2, las medidas Cantidad, Precio y Monto son aditivas, semiaditivas y derivadas, respectivamente.

Las dimensiones se utilizan para analizar medidas desde varias perspectivas. Una dimensión está compuesta ya sea por un nivel o por una o más *jerarquías* que establecen rutas de agregación significativas. Una jerarquía incluye varios niveles relacionados entre sí, como se aprecia en la Figura 3.1b. Dados dos niveles relacionados de una jerarquía, el nivel inferior se conoce como nivel *hijo* y el superior como nivel *padre*. De esta manera, las relaciones que forman jerarquías se conocen como *relaciones padre-hijo*. Debido a que se utilizan solamente para trasladarse de un nivel a otro, estas relaciones se representan simplemente con una línea para simplificar la notación.

Las relaciones padre-hijo se caracterizan por las *cardinalidades*, según se muestra en la Figura 3.1c, indicando el número mínimo y máximo de miembros en un nivel que pueden estar relacionados con un miembro en otro nivel. Por ejemplo, en la Figura 3.2 el nivel hijo Producto está relacionado con el nivel padre Categoría por medio de una cardinalidad uno a muchos, lo cual significa que cada producto pertenece a una sola categoría y que cada categoría puede tener muchos productos.

Las jerarquías en una dimensión podrían expresar varias estructuras utilizadas para fines analíticos; por ello, incluimos un *criterio de análisis* (Figura 3.1f) para diferenciarlas. Por ejemplo, la dimensión Producto en la Figura 3.2 incluye dos jerarquías: Grupos producto y Distribución. La primera comprende los niveles Producto, Categoría y Departamento mientras que la segunda incluye los niveles Producto y Distribuidor. Si un nivel contiene atributos que forman una jerarquía, tales como Nombre ciudad y Nombre estado en la dimensión Tienda de la Figura 3.2, ello significa que el usuario no está interesado en utilizar esta jerarquía para fines de agregación.

Los niveles en una jerarquía sirven para analizar datos de diferente *granularidad*, es decir, niveles de detalle. Por ejemplo, el nivel Producto contiene información específica sobre productos, mientras que el nivel Categoría podría servir para verlos desde una perspectiva más general con respecto a las categorías a las cuales pertenecen. El nivel en una jerarquía que contiene la información más detallada se llama *nivel hoja* y debe ser idéntico para todas las jerarquías dentro de una dimensión. El nombre del nivel hoja se utiliza para definir el nombre de la dimensión. El último nivel en una jerarquía, que representa la información más general, se conoce como *nivel raíz*. Si una dimensión engloba varias jerarquías, sus niveles raíz podrían ser diferentes. Por ejemplo, ambas jerarquías en la dimensión Producto que aparecen en la Figura 3.2 contienen el mismo nivel hoja, Producto, y tienen, a la vez, niveles raíz diferentes, Departamento y Distribuidor.

En algunas publicaciones, la raíz de una jerarquía se representa mediante un nivel llamado *Todo*. El diseñador de la aplicación debe tomar la decisión de incluir este nivel en los esquemas multidimensionales. En este libro no presentamos el nivel *Todo* para las diversas jerarquías, pues consideramos que no sólo carece de relevancia cuando se aplica a esquemas conceptuales, sino que también aumenta innecesariamente la complejidad de éstas.

Los atributos llave de un nivel padre definen cómo se agrupan los miembros hijo. Por ejemplo, en la Figura 3.2, el Nombre departamento en el nivel Departamento es un atributo llave que se utiliza para agrupar diferentes miembros de categoría durante la navegación ascendente del nivel Categoría hasta el nivel Departamento. No obstante, en el caso de las relaciones padre-hijo de muchos a muchos, también es necesario determinar cómo distribuir las medidas de un miembro hijo a sus miembros padre. Por ejemplo, en la Figura 3.2, la relación entre Producto y Distribuidor es muchos a muchos, es decir, el mismo producto puede ser distribuido por varios distribuidores. La notación de la Figura 3.1g indica que se utiliza un factor de distribución para dividir las medidas asociadas con un producto entre sus categorías.

Por último, en algunos casos ocurre que dos o más relaciones padre-hijo son *exclusivas*, lo cual se representa con el símbolo que se muestra en la Figura 3.1h. En la Figura 3.2 se ofrece un ejemplo donde los clientes pueden ser personas o compañías. Por consiguiente, y de acuerdo con su tipo, los clientes participan en un solo camino de la relación a partir del nivel Cliente: las personas están relacionadas con el nivel Profesión, mientras que las organizaciones lo están con el nivel Sector.

3.2 Jerarquías de los almacenes de datos

Las jerarquías son fundamentales en las aplicaciones analíticas, puesto que brindan los medios para representar los parámetros que se analizan en diferentes niveles de abstracción. En situaciones de la vida real, los usuarios deben lidiar con jerarquías complejas de diversa índole. No obstante, los modelos lógicos de los sistemas de almacenes de datos y OLAP que existen en la actualidad únicamente permiten manejar un conjunto limitado de tipos de jerarquías. Por lo tanto, los usuarios a menudo no logran captar la semántica esencial de las aplicaciones multidimensionales y el análisis que realizan lo deben limitar considerando sólo el conjunto predefinido de jerarquías que ofrecen las herramientas en uso. Asimismo, los enfoques que actualmente se aplican para representar jerarquías a nivel conceptual se centran principalmente en rutas de agregación representadas a nivel de esquema; en otras palabras, se centran en establecer secuencias de niveles que deben atravesarse durante las operaciones de navegación

ascendente y descendente. No obstante, como veremos más adelante en esta sección, la diferenciación entre los varios tipos de jerarquías también debe realizarse a nivel de instancia, es decir, debemos tomar en cuenta las cardinalidades en las relaciones padre-hijo.

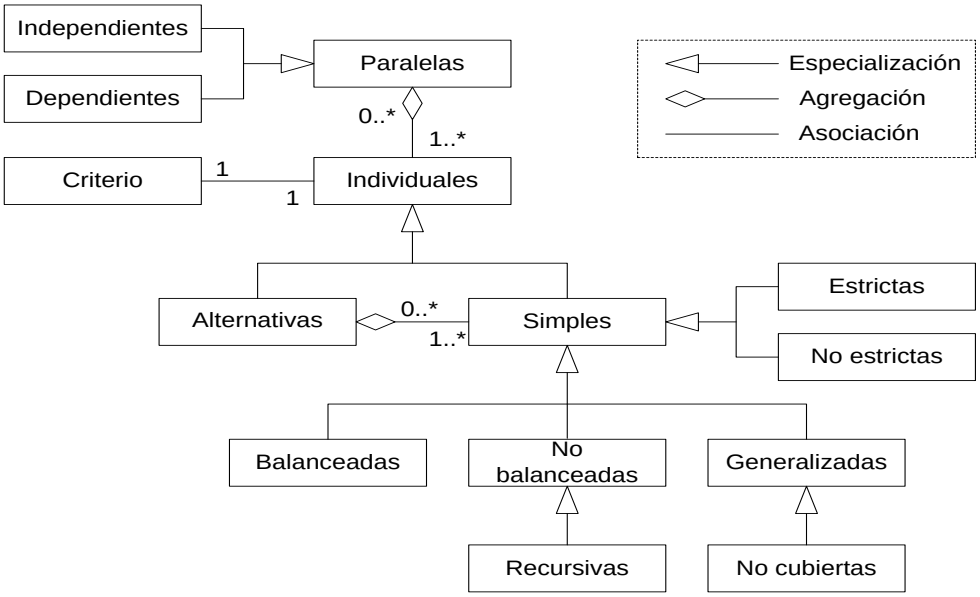


Figura 3.3. Clasificación de jerarquías

En esta sección examinamos varios tipos de jerarquías que pueden representarse en el modelo MultiDim. Con el fin de dar una visión más general de nuestra propuesta para los diversos tipos de jerarquías y las relaciones entre ellas, presentamos primero una clasificación de éstas utilizando el metamodelo que se muestra en la Figura 3.3. Como puede apreciarse en dicha figura, las jerarquías paralelas pueden especializarse en jerarquías independientes y dependientes, según compartan o no niveles comunes. Una jerarquía paralela es una agregación de jerarquías individuales y cada jerarquía individual está asociada con un criterio de análisis. Una jerarquía individual puede ser simple o alternativa; esta última está compuesta por una o varias jerarquías simples asociadas con el mismo criterio de análisis. Las jerarquías simples se pueden especializar en balanceadas, no balanceadas y generalizadas. Asimismo, las jerarquías recursivas y no cubiertas son un caso especial de jerarquías no balanceadas y generalizadas, respectivamente. Para cada una de estas jerarquías simples puede aplicarse otra especialización, dependiendo de si las cardinalidades entre los niveles padre e hijo son uno a muchos o muchos a muchos. Las primeras se conocen como *jerarquías estrictas* y las últimas como *jerarquías no estrictas*.

A continuación nos referimos de forma más detallada a los diferentes tipos de jerarquías, las cuales presentamos en los niveles conceptual y lógico. En la Sección 3.7 se explica su implementación a nivel físico con ejemplos utilizando Microsoft Analysis Services 2005 y Oracle OLAP.

3.2.1 Jerarquías simples

Las *jerarquías simples* son aquellas en las que, si todas las relaciones padre-hijo que las componen son uno a muchos, la relación entre sus miembros puede representarse en forma de árbol. Estas jerarquías utilizan un solo criterio de análisis y se dividen en balanceadas, no balanceadas y generalizadas.

Jerarquías balanceadas

Una *jerarquía balanceada* tiene una sola ruta a nivel de esquema (Figura 3.4a). A nivel de instancias, los miembros forman un árbol donde todas las ramas tienen la misma longitud (Figura 3.4b). Tal y como lo implican las cardinalidades, todos los miembros padre tienen por lo menos un miembro hijo, y un miembro hijo pertenece a un solo miembro padre. Por ejemplo, en la Figura 3.4, cada categoría tiene asignado al menos un producto, y un producto pertenece a sólo una categoría.

Jerarquías no balanceadas

Una *jerarquía no balanceada*[6] tiene una sola ruta a nivel de esquema. Sin embargo, según lo implican las cardinalidades, a nivel de instancias algunos miembros padre podrían no tener miembros hijo asociados. La Figura 3.5a muestra una jerarquía en la cual un banco está formado por varias sucursales: algunas de ellas tienen agencias con cajeros automáticos, algunas sólo tienen agencias y las sucursales pequeñas no tienen ninguna división organizacional. En consecuencia, a nivel de instancias los miembros representan un árbol no balanceado (Figura 3.5b), es decir, las ramas del árbol tienen diferentes longitudes, puesto que algunos miembros padre no tienen miembros hijo asociados. Tal como sucede en el caso de las jerarquías balanceadas, las cardinalidades implican que cada miembro hijo ha de pertenecer, por lo menos, a un miembro padre. Por ejemplo, en la Figura 3.5 cada agencia pertenece a una sucursal.

---

[6] También conocida como jerarquía heterogénea [107] o jerarquía “non-onto” [233].

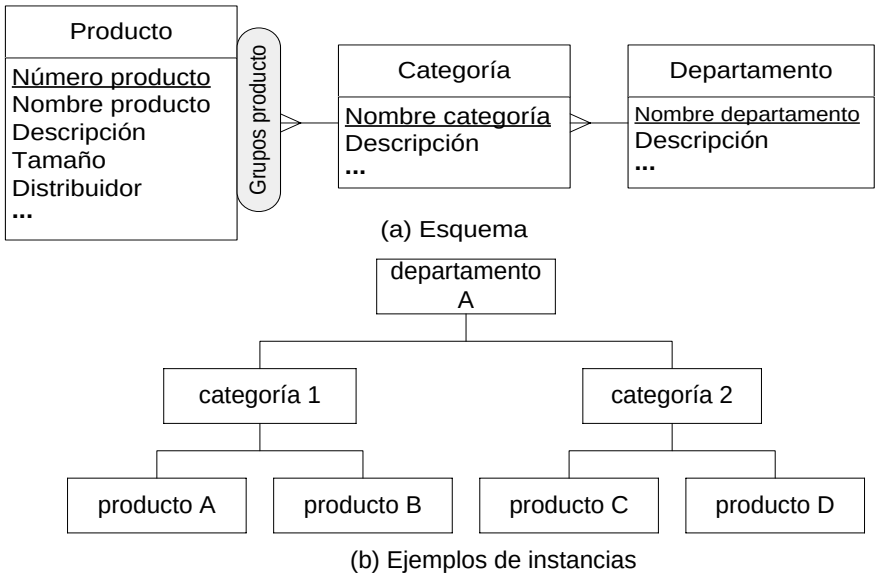


Figura 3.4. Jerarquía balanceada

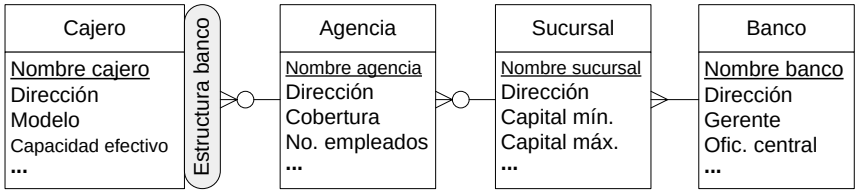
Las jerarquías no balanceadas incluyen un caso especial al cual denominamos *jerarquías recursivas*[7]. En este tipo de jerarquía el nivel está relacionado consigo mismo por medio de dos roles de una relación padre-hijo. En la Figura 3.6a se ofrece un ejemplo que representa un organigrama en términos de la relación empleado-supervisor. Los roles de subordinado y supervisor de la relación padre-hijo están enlazados con el nivel Empleado. Las jerarquías recursivas se utilizan mayormente cuando todos los niveles de jerarquía expresan la misma semántica, es decir, cuando las características de los hijos y padres son similares (o iguales), tal como se ilustra en la Figura 3.6a, donde un empleado tiene un supervisor que también es empleado.

Según se muestra en la Figura 3.6b, una jerarquía recursiva podría utilizarse para representar la jerarquía no balanceada de la Figura 3.5a. No obstante, en esta representación la semántica de la jerarquía se pierde de alguna forma, ya que los niveles correspondientes a los conceptos en diferentes granularidades se representan en conjunto, por ejemplo, cajero automático, agencia y sucursal. Vale notar que se necesita un atributo diferenciado adicional, Tipo entidad, para identificar el nivel al cual pertenecen los miembros. Además, la estructura de la jerarquía no es clara y es preciso tener acceso a todos los miembros para reconstruirla, es decir, debemos movernos hacia el nivel de instancias para acceder a la información relativa al esquema. Adicionalmente, la representación de esta jerarquía, mostrada en la Figura 3.6b, es menos expresiva que la que se presenta en la Figura 3.5a, ya que no contempla la agrupación de atributos de acuerdo con los niveles a los cuales pertenecen.

[7] También conocidas como jerarquías padre-hijo [156, 206].

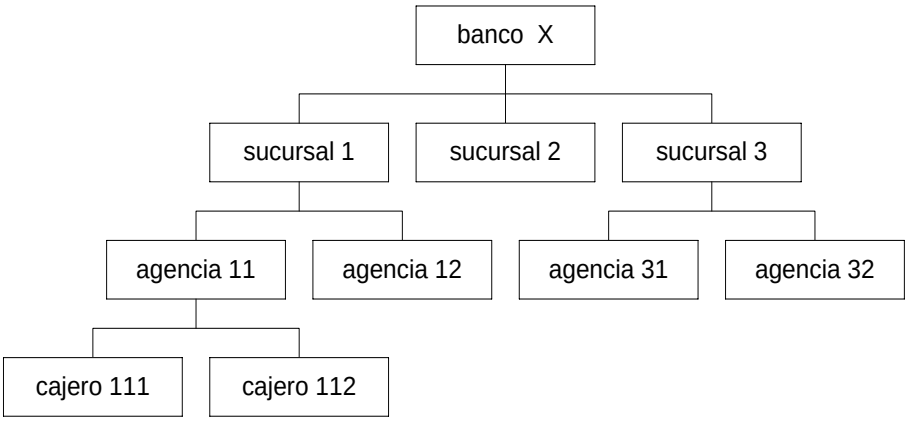
Jerarquías generalizadas

Algunas veces una dimensión incluye subtipos que pueden representarse por medio de una relación de generalización/especialización [5, 8, 161, 170]. Adicionalmente, los subtipos especializados pueden incluir su propia jerarquía. En el ejemplo que se ofrece en la Figura 3.7, el supertipo Cliente es especializado en los subtipos Persona y Compañía para distinguir los dos tipos de clientes. Asimismo, los subtipos tienen jerarquías con niveles comunes (por ejemplo, Sucursal y Cobertura) y niveles específicos (por ejemplo, Profesión y Clase para Persona, y Tipo y Sector para Compañía).



(a) Esquema

Figura 3.5. Jerarquía no balanceada



(b) Ejemplos de instancias

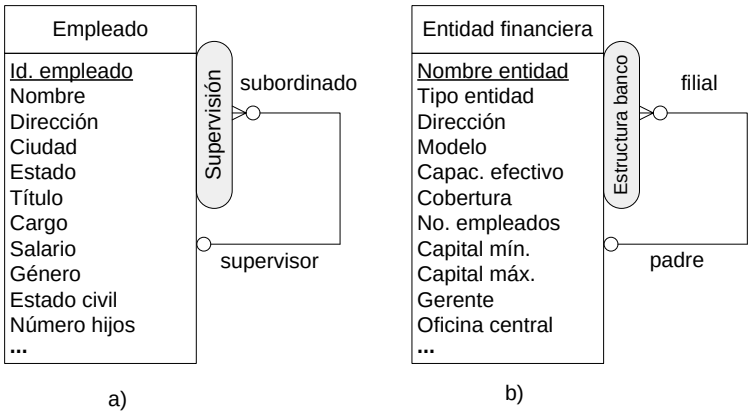


Figura 3.6. Ejemplos de jerarquías recursivas

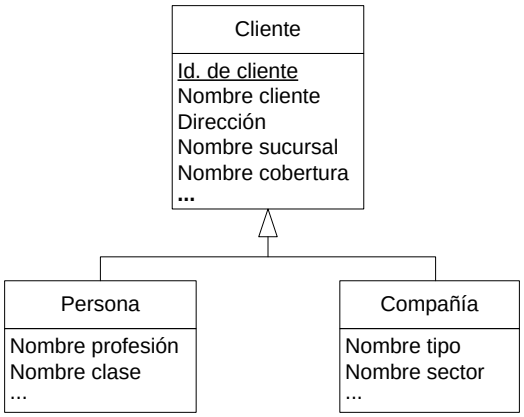


Figura 3.7. Representación Entidad-Relación de tipos de clientes

En la representación entidad-relación (ER), no se pueden distinguir claramente las rutas jerárquicas, en particular porque los niveles superiores de jerarquía (por ejemplo, Cobertura) en un supertipo no están relacionados con los otros niveles de jerarquía en sus subtipos. Por lo tanto, no es posible deducir la información relativa a los niveles que forman una jerarquía y las relaciones padre-hijo entre dichos niveles. Por ejemplo, en la Figura 3.7 no se aprecia claramente que las medidas relacionadas con un cliente que es una persona pueden agregarse utilizando una jerarquía formada por los niveles Profesión–Clase–Sucursal–Cobertura. Además, si bien ambas jerarquías pueden representarse independientemente repitiendo los niveles comunes, la complejidad del esquema se reduce si es posible incluir los niveles compartidos caracterizados por la misma granularidad, tales como los niveles de Sucursal y Cobertura de nuestro ejemplo. Para representar este tipo de jerarquías, proponemos la notación gráfica mostrada en la Figura 3.8, en la cual los niveles de jerarquía comunes y específicos, así como las relaciones padre-hijo entre dichos niveles, se representan de manera clara. A estas jerarquías las denominamos *jerarquías generalizadas*.

A nivel de esquema, una jerarquía generalizada contiene múltiples rutas exclusivas que comparten, por lo menos, el nivel hoja; también podrían compartir otros niveles según se muestra en la Figura 3.8a. Todas estas rutas representan una jerarquía y usan un mismo criterio de análisis. A nivel de instancia, cada miembro de la jerarquía pertenece a una sola ruta, tal como se puede observar en la Figura 3.8b[8]. Utilizamos el símbolo ⓧ para indicar que las rutas son exclusivas para cada miembro. Esta notación es equivalente a la anotación *xor* utilizada en el Lenguaje Unificado de Modelado (UML) [30][9]. Los niveles en los cuales las rutas alternativas se separan y unen se conocen como *niveles de separación* y *de unión*, respectivamente.

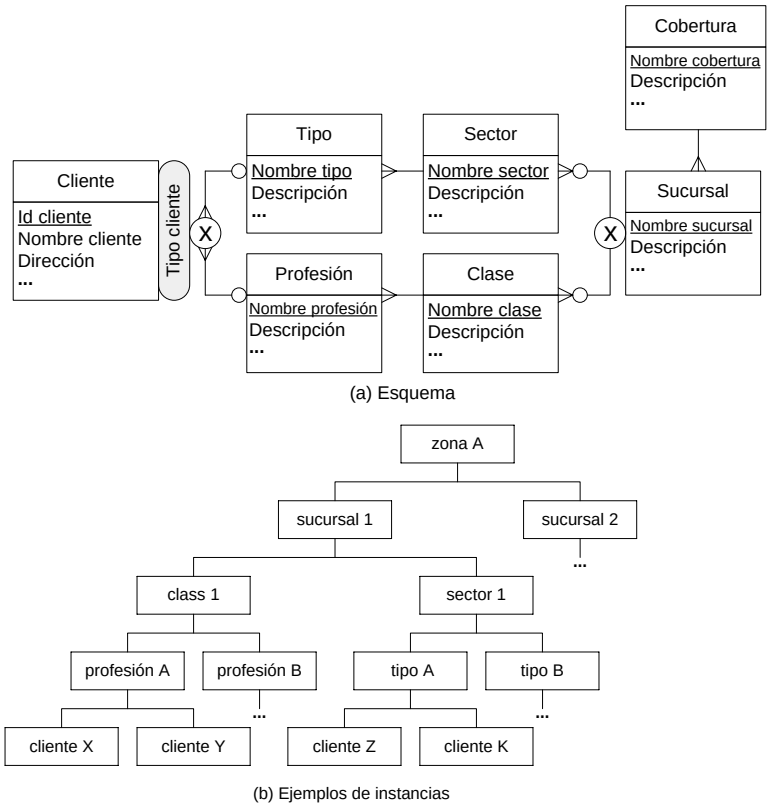


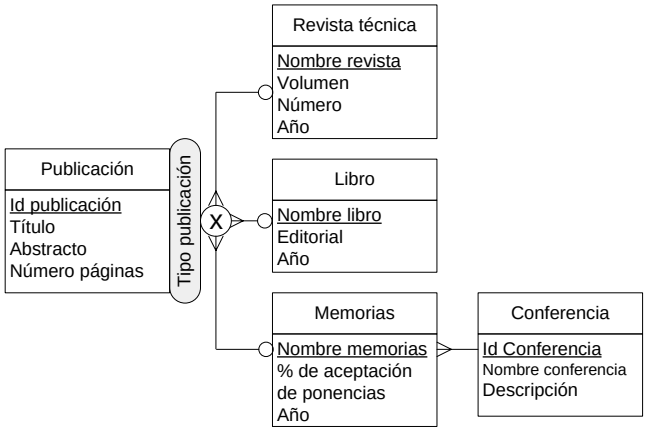
Figura 3.8. Jerarquía generalizada

En el ejemplo de la Figura 3.8a, las rutas entre los dos símbolos ⓧ se refieren a atributos que pueden usarse para propósitos de agregación para los subtipos especializados que se muestran en

[8] Nótese que, según se muestra con los puntos suspensivos, no todos los miembros están representados en la figura.

[9] Esta notación también la emplean Hurtado et al. [106] para definir las restricciones de separación.

la Figura 3.7. La ruta inferior (la que contiene los niveles Profesión y Clase) corresponde al subtipo Persona, mientras que la superior (la que contiene los niveles Tipo y Sector) indica el



subtipo Compañía.

Figura 3.9. Jerarquía generalizada sin nivel de unión

Nótese que el supertipo incluido en la relación de generalización/especialización de la Figura 3.7 se utiliza en jerarquías generalizadas para representar un nivel hoja. No obstante, abarcará únicamente aquellos atributos que representen conceptos en el nivel más bajo de granularidad (por ejemplo, Id. del cliente, Nombre cliente y Dirección). Esto es necesario para garantizar que todos los miembros de la jerarquía se puedan enlazar con la misma relación de hechos y evitar la posibilidad de que cada subtipo se maneje como una dimensión separada (por ejemplo, dimensiones separadas para los subtipos Persona y Compañía) con diferentes relaciones de hechos.

Adicionalmente, en las jerarquías generalizadas es importante hacer la distinción entre niveles de separación y unión con el fin de obtener una correcta agregación de medidas durante las operaciones de navegación ascendente y descendente [106]. Este tipo de jerarquía no satisface la condición de sumarización [164] (véase la Sección 2.6.2), porque el mapeo que se lleva a cabo desde el nivel de separación hasta los niveles padre es incompleto; por ejemplo, no todos los clientes se mueven hacia arriba para llegar al nivel Profesión. De esta manera, el mecanismo de agregación debe modificarse cuando se alcanza un nivel de separación durante una operación ascendente. Asimismo, las condiciones de sumarización también pueden variar cuando se trata de un nivel de unión. Para establecer procedimientos de agregación correctos podría aplicarse el enfoque propuesto por Hurtado et al. [106] para jerarquías heterogéneas. Por otra parte, el enfoque tradicional puede utilizarse para agregar las medidas de los niveles comunes en una jerarquía.

En las jerarquías generalizadas no es necesario que los niveles de separación se unan. Por ejemplo, la Figura 3.9 muestra una jerarquía generalizada para analizar publicaciones internacionales. Los tipos de publicación que se toman en cuenta son tres: revistas técnicas, libros y memorias de conferencias. Estas últimas pueden agregarse al nivel de conferencia; no obstante, no existe un nivel común de unión para todas las rutas.

Dentro de las jerarquías generalizadas existe un caso especial que generalmente se conoce como *jerarquías no cubiertas* o *jerarquías desiguales* [157, 233]. El ejemplo de la Figura 3.10 representa una compañía de distribución con bodegas ubicadas en distintos estados. No obstante, la división geográfica en estos estados podría variar; por ejemplo, para algunos quizá se podría omitir la división en condados. Una jerarquía no cubierta es una jerarquía generalizada con la restricción adicional de que las rutas alternativas se obtienen omitiendo uno o más niveles intermedios. A nivel de instancias, cada miembro hijo tiene únicamente un miembro padre, a pesar de que la longitud de la ruta desde las hojas hasta el mismo nivel padre puede ser distinta para miembros diferentes.

Nótese que en un modelo multidimensional no todas las jerarquías de generalización/ especialización pueden ser representadas por medio de jerarquías generalizadas. Recordemos que una generalización/especialización puede ser (a) total o parcial, y (b) disjunta o solapada [67] (véase la Sección 2.2). Las especializaciones parciales inducen una ruta adicional en la jerarquía generalizada que conecta a los niveles comunes. Por ejemplo, si la generalización en la Figura 3.7 fuera parcial —lo cual indicaría que algunos clientes no se clasificaron ni como personas ni como compañías— entonces en la jerarquía de la Figura 3.8a se necesitaría una ruta adicional entre los niveles Cliente y Sucursal.

Por otra parte, para las generalizaciones solapadas es posible tener varias opciones en función de los requerimientos del usuario y la disponibilidad de las medidas. Consideremos, por ejemplo, una generalización solapada en la cual una persona que es dueña de una empresa compra productos ya sea para uso personal o para la compañía. Si sólo se conocen las medidas para la superclase Cliente, entonces sólo se representará la jerarquía con los niveles comunes, por ejemplo, los niveles de Cliente, Sucursal y Cobertura que aparecen en la figura 3.8a. Si sólo se conocen las medidas que corresponden a cada subclase (por ejemplo para Persona y Compañía) para cada especialización se pueden crear dimensiones y relaciones de hechos separadas con sus respectivas medidas. No obstante, se requiere realizar un gran esfuerzo de programación para poder manejar las dimensiones con conjuntos solapados de miembros y diseñar procedimientos de agregación que nos permitan atravesar los niveles de jerarquía comunes. Otra solución sería inhabilitar las generalizaciones solapadas para las jerarquías multidimensionales.

3.2.2 Jerarquías no estrictas

Para las jerarquías simples descritas anteriormente, asumimos que cada relación padre-hijo tiene una cardinalidad de uno a muchos, es decir, un miembro hijo está relacionado, como máximo, con un miembro padre y un miembro padre podría estar relacionado con varios miembros hijo. Sin embargo, en las aplicaciones de la vida real es muy común encontrar relaciones muchos a muchos entre los niveles padre e hijo. Por ejemplo, un diagnóstico podría pertenecer a varios grupos de diagnóstico [233], una semana podría cubrir un período de dos meses, un celular podría clasificarse en varias categorías de producto y así sucesivamente.

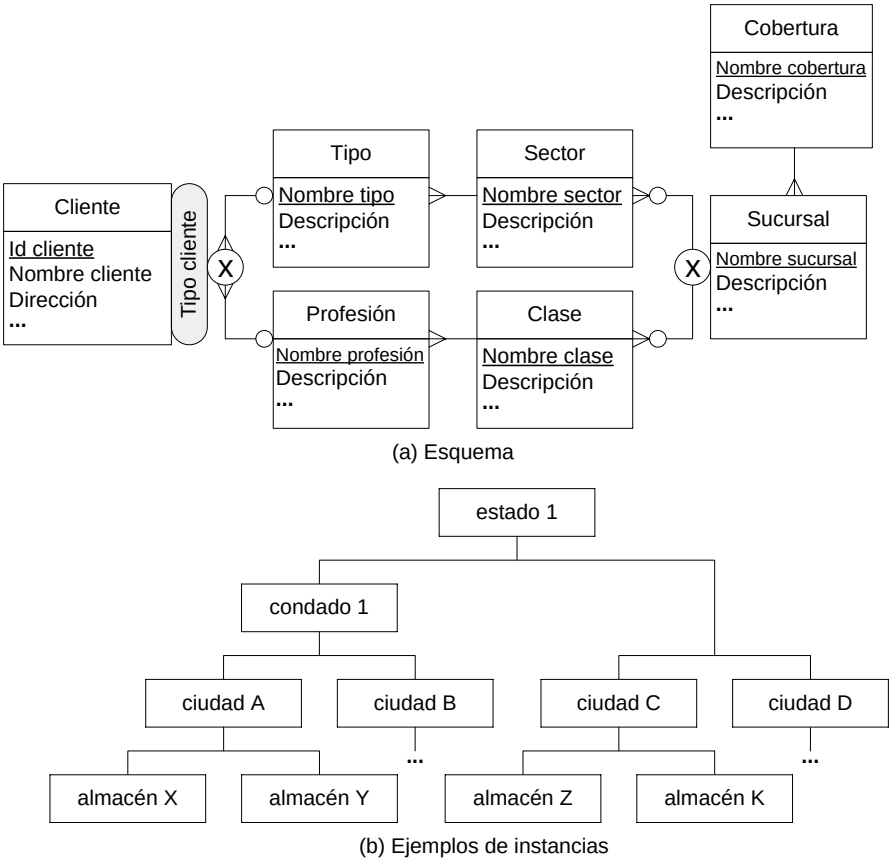


Figura 3.10. Jerarquía no cubierta

Una jerarquía es *no estricta* si, a nivel de esquema, posee al menos una relación muchos a muchos; es *estricta* si todas sus relaciones tienen cardinalidades uno a muchos. El hecho de que una jerarquía sea estricta o no es ortogonal a su tipo. De esta manera, los diferentes tipos de jerarquías descritas con anterioridad pueden ser estrictas o no estrictas.

En la Figura 3.11a se muestra una jerarquía balanceada no estricta en la cual un producto podría pertenecer a varias categorías diferentes[10]. Por ejemplo, un celular podría no sólo proveer servicio de llamadas sino también funcionar como agenda electrónica (PDA) y como reproductor de MP3. En una jerarquía estricta, este producto puede pertenecer únicamente a una categoría: teléfono. En una jerarquía no estricta podría pertenecer a tres categorías: teléfono, PDA y reproductor de MP3. Por lo tanto, puesto que a nivel de instancias un miembro hijo podría tener más de un miembro padre, los miembros de la jerarquía forman un gráfico acíclico (Figura 3.11b).

[10] Este ejemplo está basado en [104].

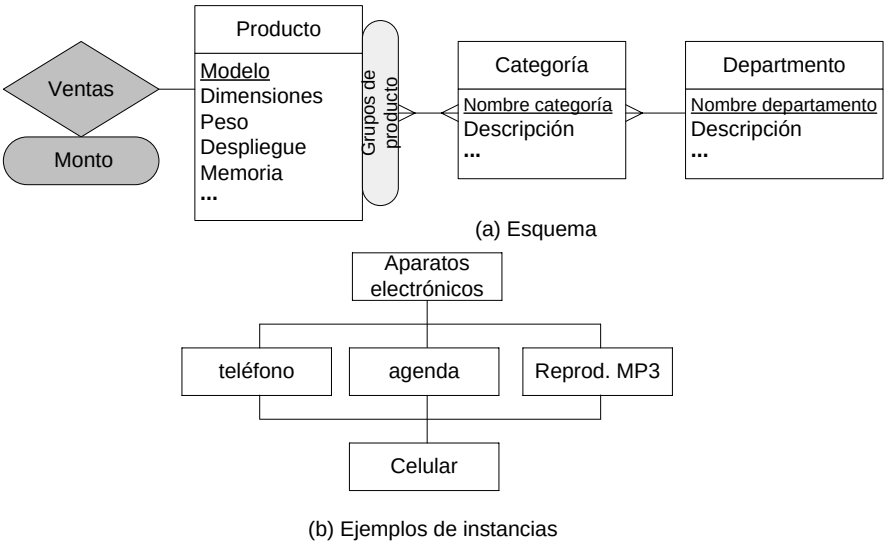


Figura 3.11. Jerarquía balanceada no estricta

Nótese el leve abuso de la terminología. Utilizamos el término jerarquía no estricta para denotar un gráfico acíclico de clasificación [104] por varias razones. Primero, el término jerarquía transmite la noción de que los usuarios necesitan analizar las medidas en función de diferentes niveles de detalle, lo cual sería menos claro si empleáramos el término gráfico acíclico de clasificación. Segundo, el término jerarquía ya lo emplean los profesionales en el campo y, como lo veremos en la próxima sección, algunas herramientas, en particular SQL Server Analysis Services 2005, permite trabajar con relaciones padre-hijo muchos a muchos. Por último, el término jerarquía también lo emplean varios investigadores (por ejemplo, [5,170, 233, 302]).

Las jerarquías no estrictas acarrean el problema del *conteo doble* de las medidas cuando una operación de navegación ascendente llega a una relación muchos a muchos. Consideremos el ejemplo de la Figura 3.12, el cual ilustra las ventas de productos con agregaciones de acuerdo con los niveles Categoría y Departamento. Supongamos que el producto con ventas iguales a 100 es un celular que también funciona como PDA y reproductor de MP3. En una jerarquía estricta, este celular puede pertenecer a una sola categoría. Por lo tanto, según se muestra en la Figura 3.12a, la suma de las ventas por categoría y departamento se puede calcular directamente. Sin embargo, quizá sería también interesante considerar el celular cuando se calcula la suma de las ventas correspondientes a PDA y reproductores de MP3. No obstante, como puede apreciarse en la Figura 3.12b, este enfoque arroja resultados agregados incorrectos, puesto que las ventas del teléfono se cuentan tres veces en lugar de una. Nótese que conforme se atraviesan los niveles jerárquicos más altos (por ejemplo, Departamento en la Figura 3.11a) se propagan resultados incorrectos.

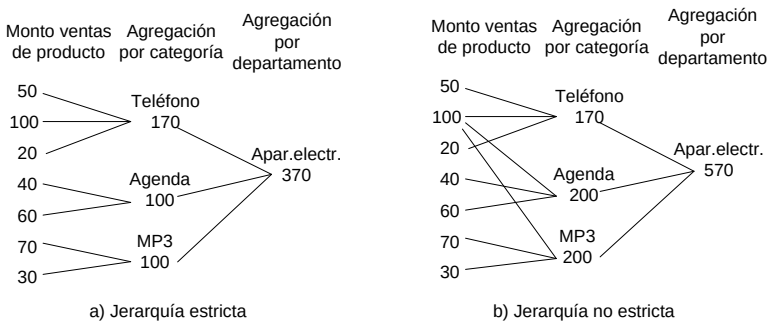


Figura 3.12. Ejemplo de agregación para una medida de monto de ventas

Una solución al problema del conteo doble de medidas, propuesta por Pedersen et al. [233], consiste en transformar una jerarquía no estricta en una jerarquía estricta creando un nuevo miembro que reúna en un solo grupo a los miembros padre que participan en una relación muchos a muchos. En nuestro ejemplo del celular, se creará un nuevo miembro de categoría que represente a las tres categorías juntas: celular, PDA y reproductor de MP3. Otra solución sería obviar la existencia de varios miembros padre y escoger uno de ellos como miembro primario [203]; por ejemplo, podríamos seleccionar sólo la categoría de celular. Sin embargo, ninguna de estas dos soluciones cumpliría los requerimientos de análisis de los usuarios, puesto que, en la primera, se incorporan categorías artificiales y, en la segunda, se pasa por alto hipótesis de análisis que son pertinentes.

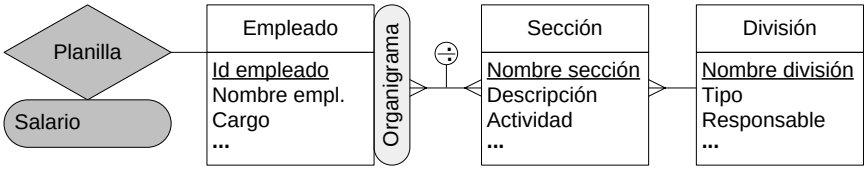


Figura 3.13. Jerarquía no estricta con un factor de distribución

El problema del conteo doble puede abordarse de forma diferente indicando cómo están distribuidas las medidas entre varios miembros padre para las relaciones muchos a muchos. Por ejemplo, la Figura 3.13 muestra una jerarquía no estricta en la cual los empleados podrían trabajar en varias secciones[11]. El esquema incluye una medida que representa el salario total de un empleado, es decir, la suma de los salarios pagados en cada sección. Podría darse el caso de que conozcamos cuál es el porcentaje de tiempo laborado en cada sección, para lo cual incluimos un símbolo ÷ adicional, , que indica que un *factor de distribución* determina cómo las medidas deben dividirse entre varios miembros padre en una relación muchos a muchos. Es importante seleccionar un factor de distribución adecuado para evitar que se obtengan resultados

[11] A efectos de simplificar, en la figura omitimos otras dimensiones, tal como la dimensión de tiempo.

aproximados. Por ejemplo, supongamos que el factor de distribución representa el porcentaje de tiempo durante el cual un empleado labora en una sección determinada. Si un empleado ocupa un cargo más alto en una sección, aunque trabaje menos tiempo en ésta, podría devengar un salario mayor. Por lo tanto, aplicar el porcentaje de tiempo como factor de distribución para las medidas que representan el salario total de un empleado podría no arrojar un resultado exacto.

Vale notar que, en el ejemplo del celular, la distribución de la medida de ventas entre tres categorías sobre la base de porcentajes previamente estipulados —por ejemplo, 70% para teléfono, 20% para PDA y 10% para MP3— quizá no brinde a los usuarios información significativa. En otras situaciones resulta imposible especificar esta distribución. En el último caso, si un usuario no requiere valores exactos de las medidas este factor de distribución puede calcularse tomando en cuenta el número total de miembros padre con el cual se asocia el miembro hijo [283]. En nuestro ejemplo del celular, en vista de que tenemos tres categorías asociadas a este teléfono, a cada categoría se le agregará un tercio del valor de la medida.

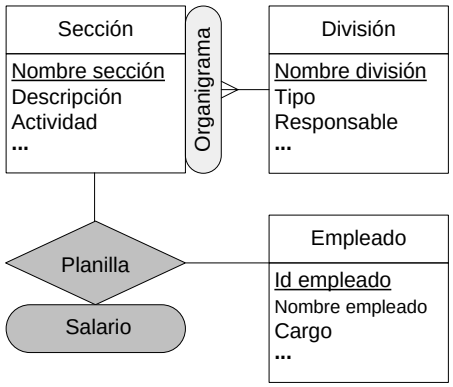


Figura 3.14. Transformación de una jerarquía no estricta en una jerarquía estricta con una dimensión adicional

Otra solución podría ser transformar una jerarquía no estricta en dimensiones independientes, tal como se muestra en la Figura 3.14. Esta solución corresponde a un esquema conceptual diferente, según el cual el enfoque del análisis cambia del salario del empleado al salario del empleado por sección. Nótese que esta solución puede aplicarse únicamente cuando conocemos la distribución exacta de las medidas, por ejemplo, cuando disponemos de los montos salariales pagados por laborar en diferentes secciones; no se puede aplicar a jerarquías no estrictas sin un factor de distribución, tal como se muestra en la Figura 3.11a.

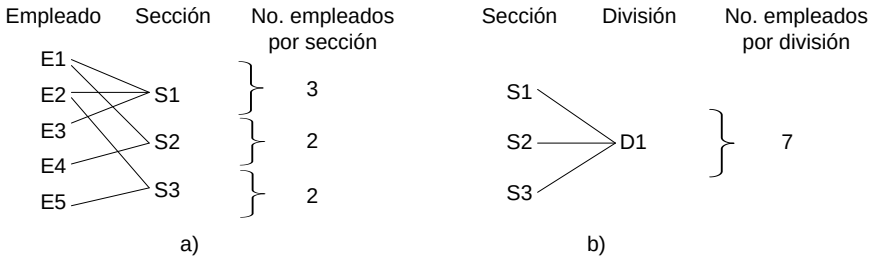


Figura 3.15. Problema de conteo doble para una jerarquía no estricta

Sin embargo, si bien la solución que se ofrece en la Figura 3.14 nos permite agregar correctamente la medida Salario, cuando aplicamos la operación de navegación ascendente del nivel Sección al nivel División, el problema del conteo doble del mismo empleado persiste. Supongamos que queremos usar el esquema de la Figura 3.14 para calcular el número de empleados por sección o por división; este valor puede calcularse contando las instancias de empleados en la relación de hechos. El ejemplo de la Figura 3.15a contempla cinco empleados asignados a varias secciones. Si contamos el número de empleados que laboran en cada sección obtendremos resultados correctos. No obstante, los valores agregados para cada sección no pueden reutilizarse para calcular el número de empleados en cada división, puesto que algunos empleados (E1 y E2 en la Figura 3.15a) se contarán dos veces y el resultado final arrojará un valor igual a 7 (Figura 3.15b) en lugar de 5.

En resumen, las jerarquías no estrictas pueden manejarse de diferentes formas, de manera que se pueda:

- Transformar una jerarquía no estricta en una estricta
  - creando un nuevo miembro padre para cada grupo de miembros padre relacionados con un miembro hijo único en una relación muchos a muchos [233];
  - escogiendo un miembro padre como miembro primario y obviando la existencia de otros miembros padre [203];
  - dividiendo la jerarquía en dos en la relación muchos a muchos, de manera que el nivel padre y los niveles subsiguientes se conviertan en una nueva dimensión.
- Incluir un factor de distribución [147].
- Calcular valores aproximados de un factor de distribución [283].

Dado que cada solución tiene sus ventajas y desventajas y requiere procedimientos de agregación especiales, el diseñador debe escoger la solución apropiada de acuerdo con la situación y los requerimientos del usuario.

3.2.3 Jerarquías alternativas

Las *jerarquías alternativas* representan una situación en la cual, a nivel de esquema, existen varias jerarquías simples no exclusivas que comparten, como mínimo, el nivel hoja y utilizan el mismo criterio de análisis. En la Figura 3.16a brindamos un ejemplo en el cual la dimensión Tiempo incluye dos jerarquías que corresponden a diferentes subdivisiones de años. Como se puede apreciar en la Figura 3.16b[12], a nivel de instancias estas jerarquías forman un grafo, puesto que un miembro hijo puede estar asociado con más de un miembro padre y estos miembros padre podrían pertenecer a diferentes niveles.

[12] Nótese que la figura no muestra los miembros del nivel Tiempo.

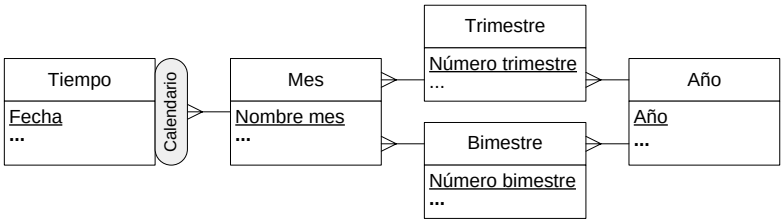
Las jerarquías alternativas se requieren cuando el usuario necesita analizar medidas desde una perspectiva determinada (por ejemplo, tiempo) empleando agregaciones alternativas. En vista de que las medidas en la relación de hechos participarán plenamente en cada jerarquía que la compone, la agregación de medidas puede llevarse a cabo para jerarquías simples. Las medidas agregadas para un nivel de jerarquía común pueden reutilizarse cuando se atraviesan las diferentes jerarquías que la componen.

Sin embargo, en el caso de las jerarquías alternativas, no es semánticamente correcto combinar en forma simultánea las diferentes jerarquías que la componen, ya que esto resultaría en intersecciones sin sentido [296], es decir, en una combinación de miembros que no tienen valores para las medidas agregadas. En el presente ejemplo, esto ocurriría al combinar los miembros B1-2001 y T2-2001 o T2-2001 y Ene-2001. Esta situación ocasiona problemas en la interpretación de los resultados de las consultas, dado que un usuario podría asumir que la falta de un valor agregado para las medidas (por ejemplo, ventas) se debe a que no hubo ventas durante un período determinado, en lugar de ver que el problema yace en una combinación inválida de niveles. Por lo tanto, los usuarios deben escoger sólo una de las rutas alternativas de agregación para su análisis y cambiar a otra si lo consideran necesario.

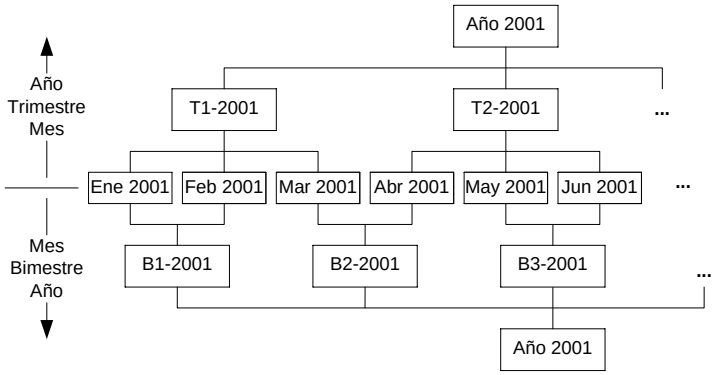
Nótese la diferencia entre jerarquías generalizadas y alternativas (Figuras 3.8 y 3.16). A pesar de que comparten algunos niveles y usan sólo un criterio de análisis, estos dos tipos de jerarquías representan situaciones diferentes. En una jerarquía generalizada, un miembro hijo está relacionado con sólo una de las rutas, mientras que en una jerarquía alternativa un miembro hijo está relacionado con todas las rutas y el usuario debe escoger una de ellas para el análisis.

3.2.4 Jerarquías paralelas

Las *jerarquías paralelas* surgen cuando una dimensión tiene varias jerarquías asociadas a ella y utiliza diferentes criterios de análisis. Dichas jerarquías pueden ser independientes o dependientes.



(a) Esquema



(b) Ejemplos de instancias

Figura 3.16. Jerarquías alternativas

Jerarquías paralelas independientes

Las *jerarquías paralelas independientes* están compuestas por jerarquías disjuntas, es decir, que no comparten niveles. Cada jerarquía componente puede ser de diferente tipo. La Figura 3.17 muestra un ejemplo de una dimensión con una jerarquía balanceada y otra no cubierta que se utilizan para diferentes criterios de análisis. La jerarquía Grupos producto se emplea para agrupar productos conforme a categorías o departamentos, mientras que la jerarquía Ubicación distribuidor los agrupa de acuerdo con las divisiones o regiones de los distribuidores.

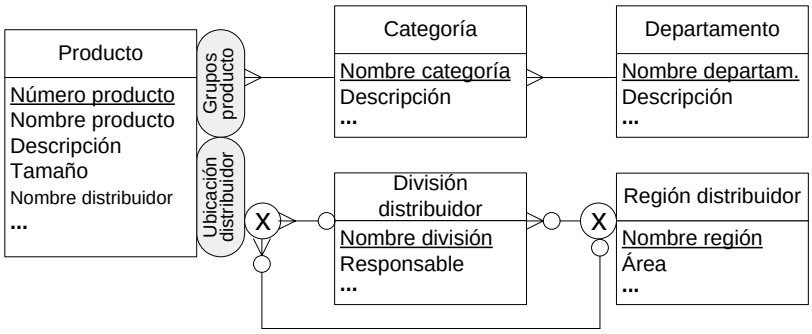


Figura 3.17. Jerarquías paralelas independientes formadas por una jerarquía balanceada y una jerarquía no cubierta

Jerarquías paralelas dependientes

Las *jerarquías paralelas dependientes* están compuestas por varias jerarquías que utilizan diferentes criterios de análisis y comparten algunos niveles. El ejemplo de la Figura 3.18 representa una compañía que requiere un análisis de ventas para tiendas ubicadas en varios países. La dimensión Tienda contiene dos jerarquías balanceadas. La jerarquía Ubicación tienda representa la división geográfica de la dirección de la tienda e incluye los niveles Tienda, Ciudad,

Estado y País. La jerarquía Organización ventas representa la división organizacional de la compañía e incluye los niveles Tienda, Distrito ventas, Estado y Región ventas. Puesto que las dos jerarquías comparten el nivel Estado, este nivel juega diferentes roles según la jerarquía seleccionada para el análisis. Compartir niveles en un esquema conceptual le permite al diseñador reducir el número de sus elementos, en otras palabras, mejorar la legibilidad sin que se pierda su semántica. Vale notar que para poder definir sin ambigüedades los niveles que componen las diferentes jerarquías, el criterio de análisis debe incluirse en el nivel que se comparte para aquellas jerarquías que continúan más allá de ese nivel.

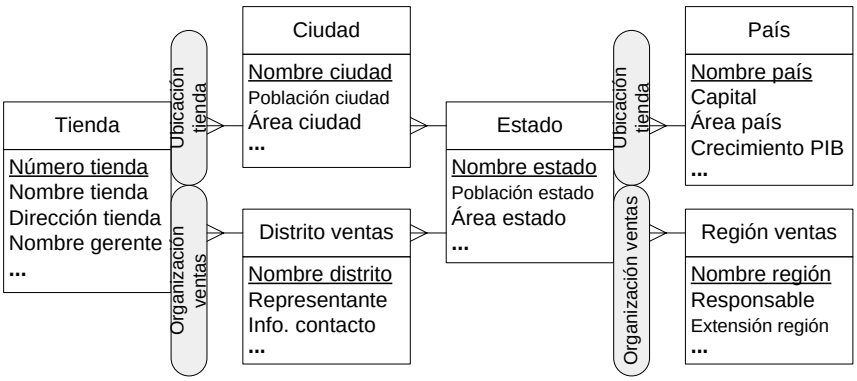


Figura 3.18. Jerarquías paralelas dependientes formadas por dos jerarquías balanceadas

Si bien tanto la jerarquía alternativa como la paralela comparten algunos niveles y podrían incluir varias jerarquías simples, ambas representan diferentes situaciones y deben distinguirse con claridad a nivel conceptual. Esto se logra incorporando un solo criterio de análisis (para las jerarquías alternativas) o varios criterios (para las jerarquías dependientes paralelas). De esta manera, el usuario puede darse cuenta de que, en el caso de las jerarquías alternativas, no tiene sentido combinar niveles de diferentes jerarquías, pero sí puede resultar útil para las jerarquías paralelas. Por ejemplo, para el esquema de la Figura 3.18, el usuario puede generar la consulta ¿Cuáles son los montos de venta para las tiendas ubicadas en la ciudad A que pertenecen al distrito de ventas X?, sin producir intersecciones irrelevantes.

Asimismo, si las jerarquías que forman parte de un conjunto de jerarquías alternativas comparten un nivel distinto del nivel hoja, entonces el miembro hijo está relacionado con el mismo nivel padre en todas las rutas jerárquicas alternativas. Por ejemplo, en el caso de la Figura 3.16, si atravesamos la jerarquía Calendario comenzando por un día específico en el nivel Tiempo siempre llegaremos al mismo año, independientemente de la ruta que se tome, es decir, puede ser Tiempo–Mes–Trimestre–Año o Tiempo–Mes–Bimestre–Año. Para las jerarquías paralelas dependientes la situación es otra. Por ejemplo, consideremos el esquema de la Figura 3.19, el cual se refiere a los lugares de residencia y trabajo de los empleados de ventas. Debería ser obvio que atravesar las jerarquías Residencia y Trabajo del nivel Empleado al nivel Estado nos permitirá llegar a diferentes estados para aquellos empleados que residan en un estado y trabajen en otro.

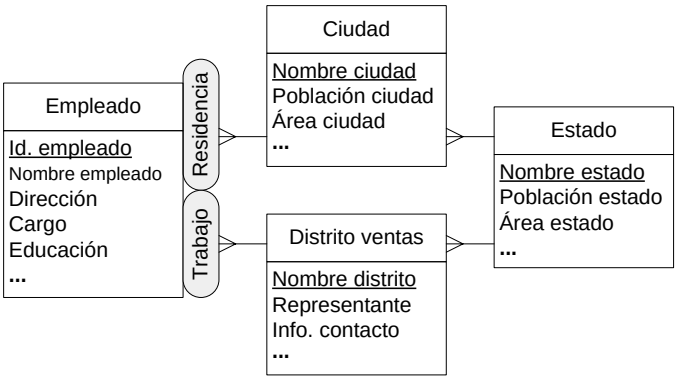


Figura 3.19. Jerarquías paralelas dependientes que conducen a diferentes miembros padre de un nivel compartido

Como resultado de la diferencia explicada anteriormente, los valores de las medidas agregadas pueden reutilizarse para los niveles compartidos en las jerarquías alternativas, lo cual no sucede para las jerarquías paralelas dependientes. Supongamos, por ejemplo, que los montos de ventas generadas por los empleados E1, E2 y E3 ascienden a €50, €100 y €150, respectivamente. Si todos los empleados residen en el estado A, pero sólo E1 y E2 trabajan en dicho estado, al agregar las ventas de todos los empleados al nivel Estado siguiendo la jerarquía Residencia obtendremos un monto total de €300, en tanto que el valor correspondiente será igual a €150 cuando se atraviesa la jerarquía Trabajo. Vale notar que ambos resultados son correctos, pues ambas jerarquías representan diferentes criterios de análisis.

3.3 Aspectos avanzados de modelado

En esta sección analizamos algunas características particulares del modelo MultiDim con el fin de mostrar su aplicabilidad para representar esquemas multidimensionales complejos. Empezamos por describir algunas jerarquías complejas que pueden representarse mediante el modelo. Posteriormente, examinaremos las dimensiones que juegan diferentes roles, las dimensiones de hechos y las dimensiones de valores múltiples.

3.3.1 Modelado de jerarquías complejas

El ejemplo de la Figura 3.20 incluye todos los tipos de jerarquías descritos en las secciones anteriores. Este ejemplo muestra tres relaciones de hechos e indica que el enfoque del análisis se centra en las ventas, la planilla y los programas de incentivos de ventas. En la figura se muestran únicamente los nombres de los niveles y se excluyen los nombres de los atributos.

Como se muestra en la figura, el modelo MultiDim nos permite representar dimensiones y niveles compartidos. Cuando se comparten niveles podemos reutilizar los datos existentes, mientras que si se comparten dimensiones se pueden analizar las medidas de diferentes relaciones de hechos utilizando la operación de navegación transversal (véase la Sección 2.6.3) [147]. Por ejemplo, el nivel Mes lo comparten dos jerarquías (Tiempo ventas y Tiempo planilla), mientras que el nivel Estado lo comparten tres jerarquías (Organización ventas, Ubicación tienda y Ubicación cliente). La dimensión Tienda, con las dos jerarquías Organización ventas y Ubicación tienda, la comparten las relaciones de hechos Ventas y Planilla. Así, la medida Monto (de la relación de hechos Ventas) puede analizarse junto con Salario base (de la relación de hechos Planilla) para diferentes tiendas y para diferentes períodos de tiempo.

Por otra parte, el modelo MultiDim nos permite definir varias relaciones padre-hijo entre los mismos niveles, por ejemplo las relaciones entre Empleado y Sección para las jerarquías Trabajo y Afiliación. Cada relación corresponde a un criterio de análisis distinto y tiene un conjunto asociado de instancias.

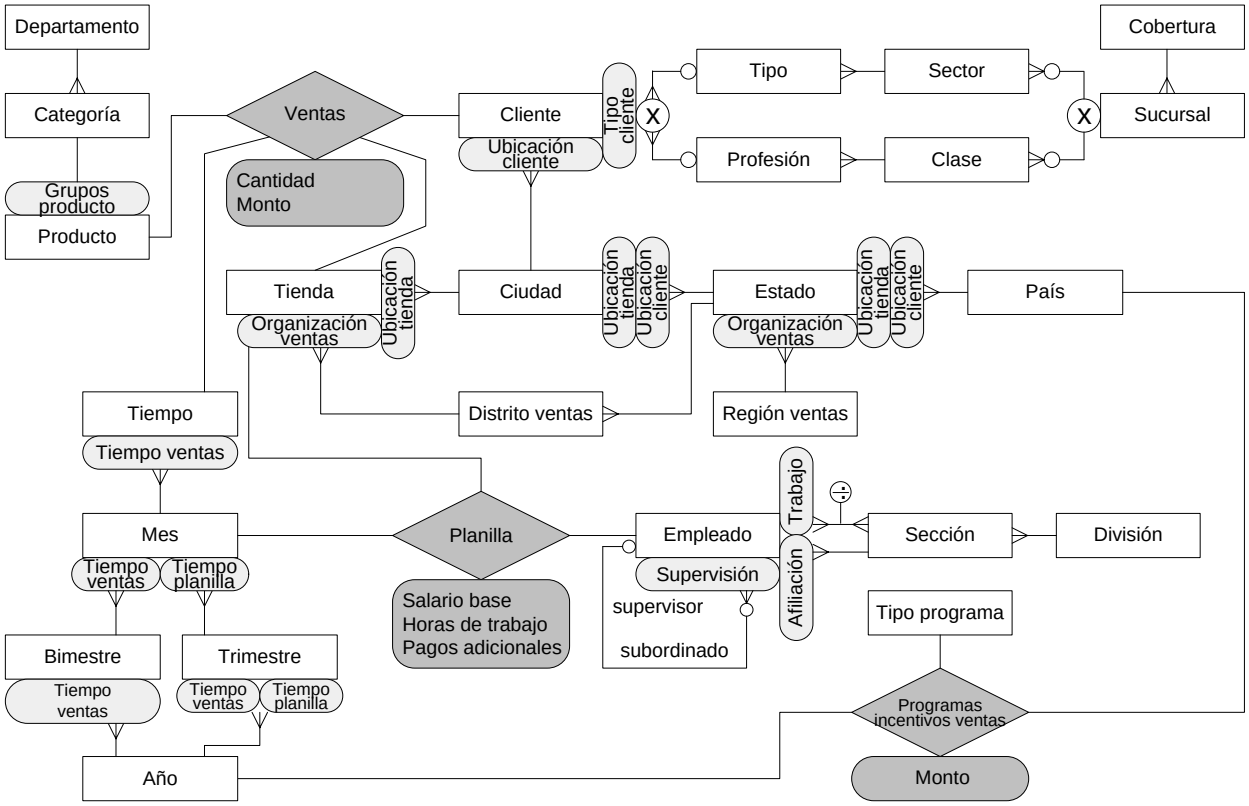


Figura 3.20. Esquema multidimensional con varios tipos de jerarquías

Otra característica del modelo MultiDim es la posibilidad de representar *jerarquías generalizadas*, las cuales, hasta donde sabemos, no se toman en cuenta en otros modelos con notaciones gráficas (por ejemplo, el modelo presentado en [110]). El ejemplo que se muestra en la Figura 3.21a (basado en [106]) modela una situación en la cual las tiendas pertenecen a varias provincias pero sólo algunas de éstas se agrupan en estados, como puede apreciarse en el nivel de instancia de la Figura 3.21b. La cardinalidad (0,1) de la Figura 3.21a indica que no a todos los miembros del nivel Provincia se les asigna un padre en el nivel Estado. Lo anterior se muestra en la Figura 3.21b, donde el miembro Todo se utiliza para representar el nivel de agregación máxima que existe para esta jerarquía. La Figura 3.22 (basada en [105]) brinda otro ejemplo de una jerarquía generalizada compleja. Ésta indica tres rutas diferentes de agregación que existen para los productos: Producto–Marca, Producto–Marca–Categoría–Departamento y Producto–Categoría–Departamento.

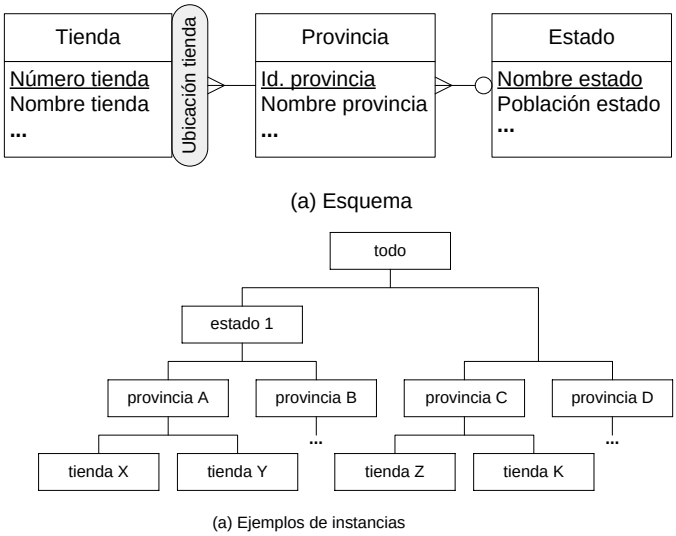


Figura 3.21. Jerarquía generalizada con diferentes niveles raíz

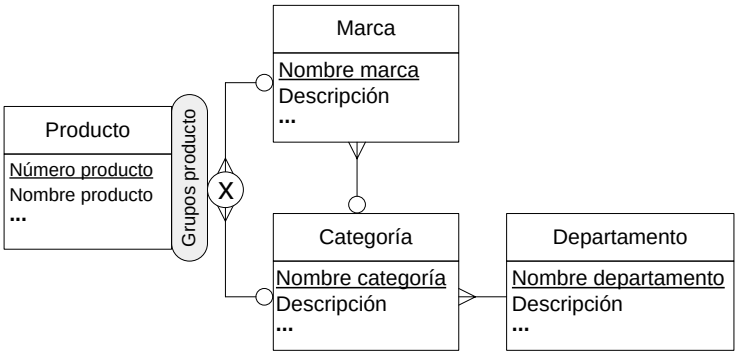


Figura 3.22. Jerarquía generalizada con tres rutas de agregación y diferentes niveles raíz

3.3.2 Dimensiones que juegan diferentes roles

En el modelo MultiDim, una relación de hechos representa una relación *n*-aria entre niveles hoja. Un miembro hoja participa en una relación de hechos de cero a muchas veces. Por ejemplo, en la Figura 3.20, si un producto aún no se ha vendido no se incluirá en la relación de hechos Ventas; por otro lado, si un producto se vendió en diferentes tiendas o en fechas diferentes, participará varias veces en la relación de hechos y en cada ocasión utilizará los valores correspondientes a las medidas Cantidad y Monto.

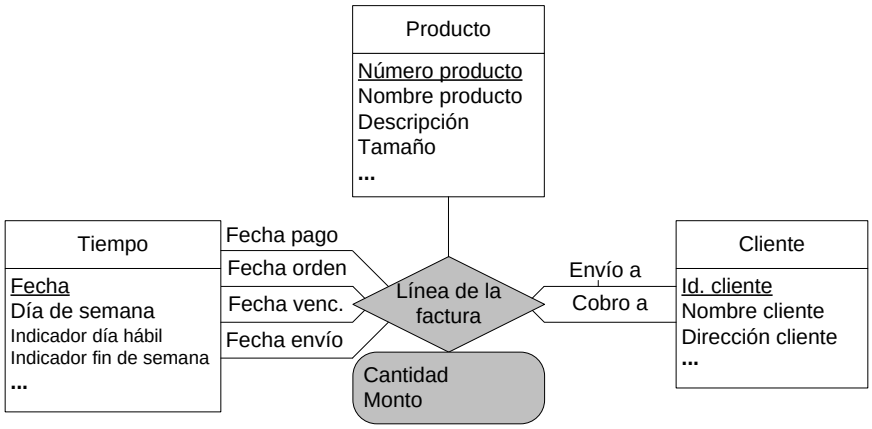


Figura 3.23. Ejemplo de una dimensión que juega diferentes roles

Como se mencionó al inicio de este capítulo, en la Sección 3.1, en algunas situaciones un nivel podría jugar diferentes roles en una relación de hechos [101, 170]. Para indicar estos diferentes roles, en el esquema incluimos enlaces adicionales entre el nivel hoja y la relación de hechos, cada uno con el correspondiente nombre de rol. El ejemplo de la Figura 3.23 nos permite analizar cuándo un producto es ordenado por el cliente (Fecha orden), cuándo se envía (Fecha envío), cuándo debe ser entregado (Fecha venc.) y cuándo se paga por él (Fecha pago). Asimismo, puesto que un producto puede ser enviado a un cliente y pagado por otro, la dimensión Cliente también juega dos roles diferentes, es decir, Envío a y Cobro a.

El esquema de la Figura 3.23 podría incluir cuatro dimensiones de tiempo específicas que indican las fechas de orden, envío, vencimiento y pago, y dos dimensiones específicas referidas a los clientes a quienes se les envían los productos y los clientes que pagan por ellos. No obstante, como puede apreciarse en la figura, la complejidad del esquema se reduce si, en lugar de duplicar las dimensiones, utilizamos los diferentes roles que las dimensiones podrían jugar.

El programa Microsoft Analysis Services 2005 [101, 157] permite manejar este tipo de dimensiones que juegan diferentes roles.

3.3.3 Dimensiones de hechos

A continuación nos referiremos a las *dimensiones de hechos* [157] o *dimensiones degeneradas* [147, 170]. En la Figura 3.24 se muestra un ejemplo típico en el cual los usuarios necesitan analizar las ventas de productos con un nivel mínimo de granularidad, representado por la línea de transacción (u orden). Vale notar que las relaciones de hechos de la Figura 3.24 difieren de la relación de hechos Ventas de la Figura 3.20, puesto que en esta última las ventas diarias de los productos se suman antes de incluirlas en la relación de hechos.

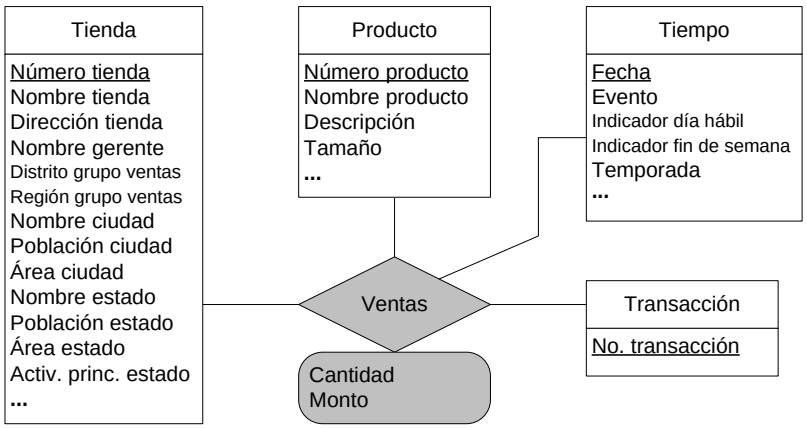


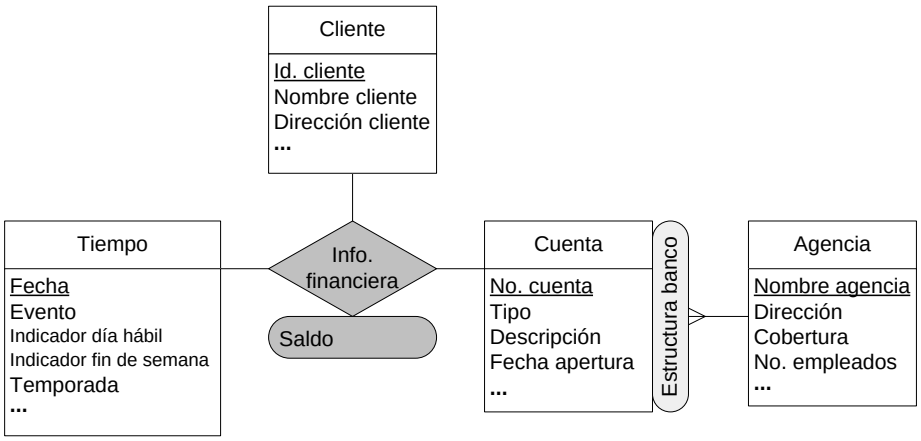
Figura 3.24. Esquema con una dimensión de hechos

Como se aprecia en la Figura 3.24, la dimensión Transacción incluye únicamente el número de transacción, puesto que otros datos, tales como la fecha de transacción o el número de tienda ya están incorporados en otras dimensiones. En la representación lógica de este esquema, esta dimensión comúnmente se convierte en un atributo de la relación de hechos. Como consecuencia, a esta dimensión se le denomina dimensión degenerada, puesto que no posee ningún otro atributo que la describa o una dimensión de hechos, puesto que se incluye como parte de una relación de hechos y, a la vez, juega el rol de una dimensión, es decir, se utiliza para propósitos de agrupación.

Consideramos que la inclusión del atributo de la dimensión en la relación de hechos debe realizarse a nivel lógico o de implementación. El esquema conceptual debe mantener esta dimensión para indicar que podría ser necesario agrupar de acuerdo con los miembros de la dimensión, es decir, según el número de transacción.

3.3.4 Dimensiones de valores múltiples

Los términos dimensiones de valores múltiples [147] y relaciones muchos a muchos entre hechos y dimensiones [170, 233, 283] se han utilizado para representar la situación en la cual varios miembros de una dimensión participan en la misma instancia de una relación de hechos. Un ejemplo común utilizado en este caso es el análisis de los saldos de los titulares de cuentas, según



se muestra en la Figura 3.25.

Figura 3.25. Esquema multidimensional para el análisis de cuentas bancarias

En vista de que una cuenta puede pertenecer a varios clientes, la agregación del saldo realizada según el cliente sumará el saldo tantas veces cuántos titulares de cuentas existan. Consideremos el ejemplo de la Figura 3.26. En algún momento en el tiempo T1, tenemos dos cuentas, A1 y A2, con saldos de 100 y 500, respectivamente. Como se aprecia en la figura, varios clientes comparten ambas cuentas: la cuenta A1 entre C1, C2 y C3, y la cuenta A2 entre C1 y C2. El saldo total de las dos cuentas es igual a 600; no obstante, la agregación (por ejemplo, según la dimensión de Tiempo o Cliente) arroja un valor igual a 1.300.

Tiempo	Cuenta	Cliente	Saldo
T1	A1	C1	100
T1	A1	C2	100
T1	A1	C3	100
T1	A2	C1	500
T1	A2	C2	500

Figura 3.26. Ejemplo de un problema de conteo doble para una dimensión de valores múltiples

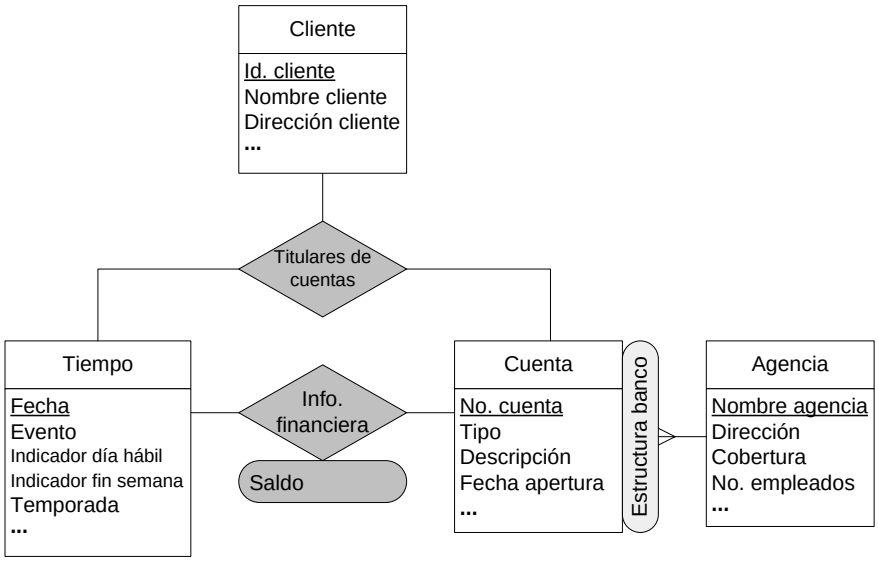
Podemos analizar el problema del *conteo doble* examinando las *formas normales multidimensionales* (FNM) definidas por Lehner et al. [163] y Lechtenbörger y Vossen [161]. Las FNM se centran en la sumarización (véase la Sección 2.6.2) y determinan condiciones que garantizan la correcta agregación de medidas en presencia de jerarquías generalizadas (véase la Sección 3.2.1). Las FNM se determinan utilizando una generalización de dependencias funcionales que toman en cuenta valores nulos. La primera forma normal multidimensional (1FNM) establece cuatro condiciones. Las primeras tres tienen como objetivo garantizar que las dependencias funcionales contenidas en un esquema reflejen aquellas que se encuentran en el dominio de aplicación. La última condición requiere que cada medida sea identificada de manera única por el conjunto de niveles hoja. El 1FNM es la base para diseñar un esquema correctamente y se utiliza para definir las FNM restantes. La segunda y tercera FNM se centran en establecer condiciones para las jerarquías generalizadas. Aquí omitimos sus descripciones ya que nuestro modelo es más general que el propuesto por Lechtenbörger y Vossen [161], es decir, el modelo MultiDim incluye jerarquías no estrictas (Sección 3.2.2) y jerarquías paralelas dependientes (Sección 3.2.4) y permite compartir niveles entre dimensiones (como se muestra en la Figura 3.20). Además, las FNM no toman en cuenta las dependencias multivaluadas y éstas son importantes para garantizar la exactitud de los esquemas multidimensionales.

Analicemos el esquema de la Figura 3.25 en términos de la 1FNM: Necesitamos establecer las dependencias funcionales que existen entre los niveles hoja y las medidas. En vista de que el saldo depende de la cuenta específica y del momento en el que se examina, la cuenta y el tiempo son los que determinan el saldo. Por lo tanto, el esquema de la Figura 3.25 no satisface la 1FNM, dado que la medida no la determinan todos los niveles hoja y, por tanto, es necesario descomponer la relación de hechos.

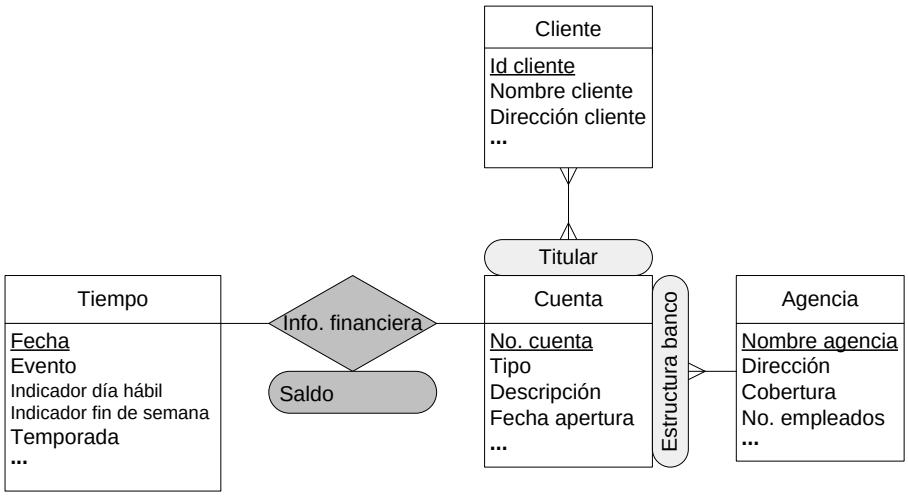
Si tomamos en consideración las dependencias multivaluadas, existen dos formas posibles de descomponer la relación de hechos Info. financiera de la Figura 3.25. En el primer caso, la misma cuenta mancomunada podría tener clientes diferentes asignados a ella durante períodos de tiempo diferentes, de manera que el tiempo y la cuenta multideterminan a los clientes. Esto nos lleva a la solución ilustrada en la Figura 3.27a, en la cual la relación original de hechos se divide

en dos relaciones de hechos, es decir, Titulares de cuenta e Info. financiera. Por otro lado, el segundo caso surge cuando la composición de los clientes que tienen cuentas mancomunadas no varía con el tiempo, de manera tal que la cuenta es la única que multidetermina a los clientes. En este caso, debemos eliminar el enlace que une al nivel Tiempo con la relación de hechos Titulares de cuenta, en la Figura 3.27a. Como alternativa, esta situación se puede modelar con una jerarquía no estricta, tal como se muestra en la Figura 3.27b.

Si bien las soluciones propuestas en la Figura 3.27 eliminan el problema del conteo doble, los dos esquemas de la Figura 3.27 requieren llevar a cabo un esfuerzo de programación para aquellas consultas que solicitan información acerca de clientes individuales. La diferencia yace en que en la Figura 3.27a es necesario realizar una operación de navegación transversal (véase la Sección 2.6.3) entre las dos relaciones de hechos, mientras que en la Figura 3.27b se deben emplear procedimientos especiales de agregación en las jerarquías no estrictas. En el caso de la Figura 3.27a, en vista de que las dos relaciones de hechos representan diferentes granularidades, las consultas con las operaciones transversales son complejas, lo cual exige que se realice una conversión de una granularidad detallada a una global (por ejemplo, agrupar clientes para saber quién posee un saldo específico en una cuenta) o viceversa (por ejemplo, distribuir un saldo entre diferentes titulares de cuentas). Vale notar también que los dos esquemas de la Figura 3.27 podrían representar la información relativa al porcentaje de participación de los clientes en las cuentas (si se conoce). Esto podría representarse mediante una medida en la relación de hechos Titulares de cuenta en la Figura 3.27a, y por un factor de distribución en la relación muchos a muchos de la Figura 3.27b.



a) Creación de dos relaciones de hechos



(b) Inclusión de una jerarquía no estricta

Figura 3.27. Dos posibles descomposiciones de la relación de hechos de la Figura 3.25

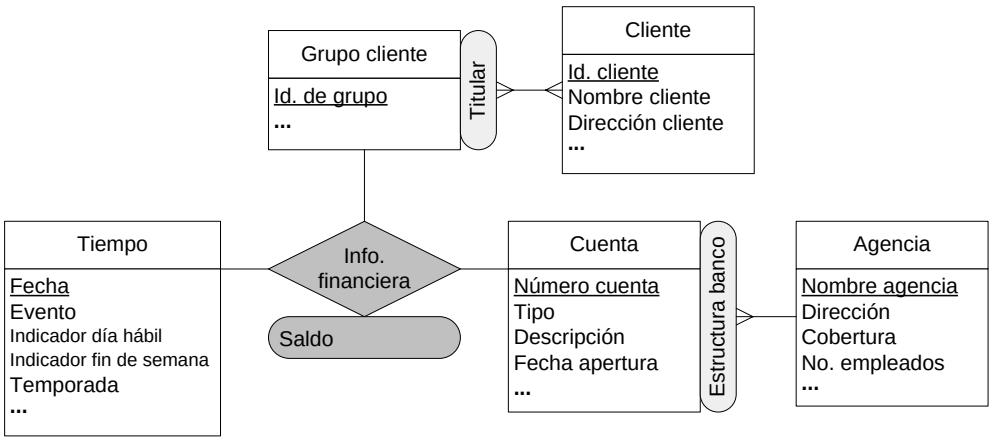


Figura 3.28. Descomposición alternativa del esquema mostrado en la Figura 3.25

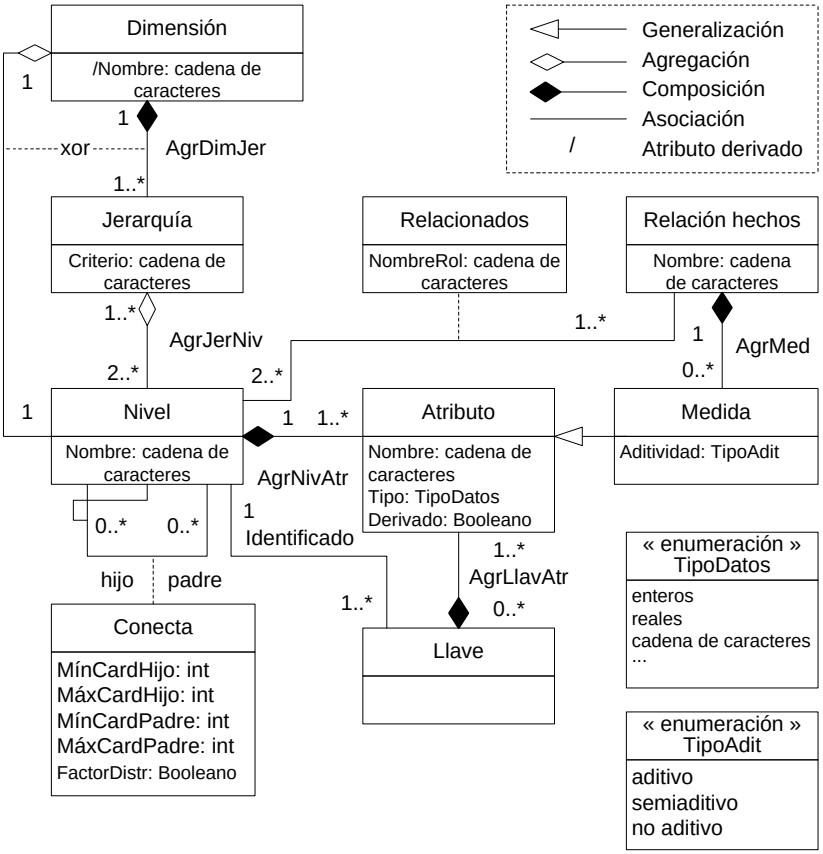
Otra solución a este problema, presentada de forma implícita por Pedersen [232] y Song et al. [283], se muestra en la Figura 3.28. En esta solución se crea un nivel adicional, el cual representa a los grupos de clientes que participan en cuentas mancomunadas. Para nuestro ejemplo de la Figura 3.26 creamos dos grupos: uno que incluye los clientes C1, C2 y C3 y otro con los clientes C1 y C2. La diferencia entre la Figura 3.28 y la solución de Pedersen consiste en el hecho de que la última abarca tanto a los grupos de clientes como a los clientes individuales, es decir, el mismo nivel contendrá miembros con diferentes granularidades de datos. Sin embargo, el esquema de la Figura 3.28 no está en la 1FNM, ya que la medida Saldo no la determinan todos los niveles, sino únicamente Tiempo y Cuenta. Por lo tanto, el esquema debe descomponerse para elaborar esquemas similares a los de la Figura 3.27, con la diferencia de que, en este caso, al nivel Cliente en ambos esquemas de la Figura 3.27 lo reemplaza una jerarquía no estricta compuesta por los niveles Grupo cliente y Cliente.

Otra solución para evitar que surjan dimensiones de valores múltiples es escoger a un cliente como titular principal de la cuenta e ignorar a los demás. De esta manera, sólo un cliente estará relacionado con un saldo específico y el esquema de la Figura 3.25 puede utilizarse para eludir el problema del conteo doble de medidas. Sin embargo, es posible que esta solución no represente una situación de la vida real y podría excluir del análisis a los otros clientes de las cuentas mancomunadas.

En resumen, las dimensiones de valores múltiples en esquemas multidimensionales pueden evitarse si aplicamos una de las soluciones presentadas en la Figura 3.27. La elección entre estas alternativas está supeditada a las dependencias funcionales y multivaluadas que existen en la relación de hechos, los tipos de jerarquías que admite el esquema y la complejidad de la implementación.

3.4 Metamodelo del Modelo MultiDim

En esta sección presentamos el metamodelo del modelo MultiDim utilizando la notación del lenguaje UML, tal como se muestra en la Figura 3.29.



3.29. Metamodelo del modelo MultiDim

Una dimensión está compuesta ya sea por un nivel o por una o más jerarquías, y cada jerarquía pertenece a una sola dimensión. Una jerarquía contiene dos o más niveles relacionados que pueden ser compartidos por diferentes jerarquías. Un nombre de criterio identifica a una jerarquía y cada nivel tiene un nombre único. El nombre del nivel hoja se utiliza para derivar el nombre de una dimensión: esto se representa mediante el atributo derivado Nombre en la Dimensión. Los niveles incluyen atributos, algunos de los cuales son atributos llave empleados para propósitos de agregación, mientras que otros son atributos descriptivos. Los atributos tienen un nombre y un tipo, el cual debe ser un tipo de dato, es decir, entero, real o cadena de caracteres. Asimismo, un atributo puede ser derivado.

Los niveles que forman jerarquías están relacionados mediante la clase de asociación Conecta. Esta clase de asociación está caracterizada por las cardinalidades mínimas y máximas de los roles padre e hijo y por el factor de distribución, si lo hubiera. Una restricción que no se muestra en la figura establece que una relación padre-hijo tiene un factor de distribución únicamente si las cardinalidades máximas del hijo y el padre son iguales a muchos.

Una relación de hechos representa a una asociación *n*-aria entre los niveles hoja con n > 1. Puesto que los niveles hoja pueden jugar diferentes roles en esta asociación, el nombre del rol se incluye en la clase de asociación Relacionado. Una relación de hechos podría contener atributos, los cuales se denominan comúnmente medidas: éstas pueden ser aditivas, semiaditivas o no aditivas.

3.5 Mapeo a los modelos relacionales y objeto-relacionales

3.5.1 Fundamentos

Los modelos conceptuales se pueden implementar utilizando varios modelos lógicos. En esta sección presentamos las reglas de mapeo que permiten implementar esquemas conceptuales diseñados utilizando el modelo MultiDim en los modelos relacionales y objeto-relacionales.

Conforme se explicó en la Sección 2.7, los almacenes de datos usualmente se implementan en bases de datos relacionales, tales como los esquemas de estrella o de copo de nieve. Existen varias razones que justifican el uso de una implementación relacional para los modelos multidimensionales.

- Las bases de datos relacionales usan estrategias muy conocidas para almacenar datos y están bien estandarizadas, no dependen de herramientas y son de fácil acceso. Por otra parte, no existe consenso en la comunidad de investigación acerca de cómo definir un nivel lógico para los sistemas OLAP [8]. Asimismo, los sistemas OLAP multidimensionales (MOLAP) difieren mucho con respecto a las estructuras utilizadas para el almacenamiento de datos.
- Se han realizado muchos estudios en el campo de las bases de datos relacionales sobre cómo mejorar la optimización de las consultas, la indexación, las operaciones de mezcla y la materialización de las vistas, considerando las particularidades de las aplicaciones de los almacenes de datos.
- Si bien los sistemas MOLAP ofrecen un mejor desempeño durante las operaciones de navegación ascendente, muchos sistemas comerciales y de prototipo utilizan el almacenamiento ROLAP (OLAP relacional) u HOLAP (OLAP híbrido, una combinación de ROLAP y MOLAP) para manejar grandes volúmenes de datos, según lo descrito en la Sección 2.7.
- Los sistemas de bases de datos relacionales comerciales [14, 101, 216] incluyen extensiones para representar y gestionar la representación multidimensional de datos.
- Las bases de datos relacionales son una capa de recursos aceptada por el **Modelo Común de Almacenes de Datos** (Common Warehouse Model – CWM) [209] del Grupo de Gestión de Objetos (Object Management Group – OMG)[13].

El modelo objeto-relacional se emplea cada vez más en las aplicaciones de bases de datos convencionales debido, entre otras razones, a que preserva los conceptos básicos del modelo relacional y, a la vez, organiza los datos utilizando un modelo orientado al objeto [155] (véase la Sección 2.3.2). Asimismo, muchos sistemas de administración de base de datos (Database Management System – DBMS) manejan el modelo objeto-relacional. En vista de que los almacenes de datos son bases de datos específicas con fines analíticos, también pueden implementarse aplicando el modelo objeto-relacional.

Los modelos de implementación que hemos escogido son el modelo relacional basado en el uso de llaves sustitutas [90] y el modelo objeto-relacional. En estos modelos cada fila en una tabla tiene una llave sustituta asociada, es decir, una llave artificial generada por el sistema. El usuario

[13] En la Sección 3.8 se ofrece una descripción del CWM.

no puede ver ni modificar los valores sustitutos. La razón por la cual se utilizan llaves sustitutas en lugar de emplear las llaves definidas por el usuario es que los almacenes de datos utilizan, por lo general, llaves generadas por el sistema para garantizar un mejor desempeño durante las operaciones de mezcla y para no depender de los sistemas transaccionales. Además, las llaves sustitutas no varían en el tiempo, de manera que dos entidades que tienen llaves sustitutas idénticas representan la misma identidad, lo cual permite añadir información histórica sin ambigüedades. Adicionalmente, en algunas situaciones es necesario almacenar información sobre una entidad, ya sea antes de asignarle un valor llave controlado por el usuario (por ejemplo, agregar un estudiante a la base de datos antes de asignarle un número de identificación) o bien una vez que cese de tenerlo (por ejemplo, cuando un empleado deja de laborar en una compañía). Además, el estándar SQL:2003 [194, 195] y algunos DBMS actuales (por ejemplo, Oracle [215]) permiten manejar tablas con filas que contienen llaves sustitutas creadas automáticamente. No obstante, los modelos relacionales u objeto-relacionales que carecen de llaves sustitutas pueden también utilizarse en nuestro mapeo para aquellos DBMS que no las manejan. En este caso, los desarrolladores del sistema son responsables de crear llaves artificiales independientes del sistemas de origen.

3.5.2 Reglas de mapeo

En vista de que el modelo MultiDim se basa en el modelo ER, su mapeo al modelo relacional está fundamentado en reglas muy conocidas, tales como las descritas en la Sección 2.3.1:

Regla 1: Un nivel corresponde a un tipo de entidad en el modelo ER. Está mapeado a una tabla que contiene todos los atributos del nivel e incorpora un atributo adicional para una llave sustituta.

Regla 2: La relación padre-hijo corresponde a un tipo de relación binaria en el modelo ER. Vale recordar que, con el fin de obtener jerarquías significativas, la cardinalidad máxima del rol padre debe ser n. De esta manera, existen dos mapeos diferentes dependiendo de la cardinalidad del rol hijo.

Regla 2a: Si la cardinalidad del rol hijo es (0,1) o (1,1), es decir, la relación padre-hijo es una a muchas, la tabla que corresponde al nivel hijo se extiende con la llave sustituta del nivel padre correspondiente, es decir, en la tabla hijo hay una llave foránea que apunta hacia la tabla padre.

Regla 2b: Si la cardinalidad del rol hijo es (0,n) o (1,n), es decir, la relación padre-hijo es muchos a muchos, se crea una nueva tabla que contiene como atributos las llaves sustitutas de los niveles padre e hijo. Si la relación padre-hijo tiene un factor de distribución, se agrega un atributo adicional a la tabla para almacenar esta información.

Regla 3: Un tipo de relación de hechos corresponde a un tipo de relación *n*-aria en el modelo ER. Una relación de hechos que contiene los niveles hoja N_1, …, N_n está mapeada a una tabla que incluye como atributos las llaves sustitutas de los niveles participantes, uno para cada rol que cada nivel juega. Además, cada atributo (es decir, medida) de la relación de hechos está mapeado a un atributo de la tabla.

Si aplicamos las reglas anteriores al ejemplo de la Figura 3.2 obtenemos como resultado las tablas que se muestran en la Figura 3.30. La tabla Ventas incluye cinco llaves foráneas, es decir, una para cada nivel participante y tres atributos que representan las medidas, es decir, Cantidad, Precio y Monto. Nótese que cuando un nivel juega varios roles, como en el caso de Tiempo, cada

rol está representado por una llave foránea; en otras palabras, la Fecha orden y la Fecha pago son ambas llaves foráneas de la dimensión Tiempo. Vale notar también que la relación padre-hijo muchos a muchos entre Producto y Distribuidor está mapeada a la tabla DistProd que contiene dos llaves foráneas.

A continuación presentamos un grupo de comandos SQL DDL que definen un extracto del esquema relacional de la Figura 3.30.

```
create table Ventas as (
        LlaveF producto integer not null,
        LlaveF tiempo orden integer not null,
        LlaveF tiempo pago integer not null,
        LlaveF tienda integer not null,
        LlaveF cliente integer not null,
        Cantidad integer not null,
        Precio decimal(7,2) not null,
        Monto decimal(7,2) not null,
        primary key (LlaveF producto, LlaveF tiempo orden , LlaveF tiempo pago,
                LlaveF tienda, LlaveF cliente),
        foreign key LlaveF producto references Producto(Llave producto),
        foreign key LlaveF tiempo orden references Tiempo(Llave tiempo),
        foreign key LlaveF tiempo pago references Tiempo(Llave tiempo),
        foreign key LlaveF tienda references Tienda(LlaveTienda),
        foreign key LlaveF cliente references Cliente(Llave cliente) );
create table Producto as (
        Llave producto integer primary key not null,
        Número producto character varying(9) not null,
        Nombre producto character varying(30) not null,
        Descripción text not null,
        Tamaño integer not null,
        LlaveF categoría integer not null,
        ...
        foreign key LlaveF categoría references Producto (Llave producto) );
create table Categoría as (
```

Llave categoría integer primary key not null,

Nombre categoría character varying(20) not null,

Descripción text not null,

LlaveF departamento integer not null,

...

foreign key LlaveF departamento references Departmento(Llave departamento));

create table Departmento as (

Llave departamento integer primary key,

Nombre departamento character varying (30) not null,

Descripción text not null,

...);

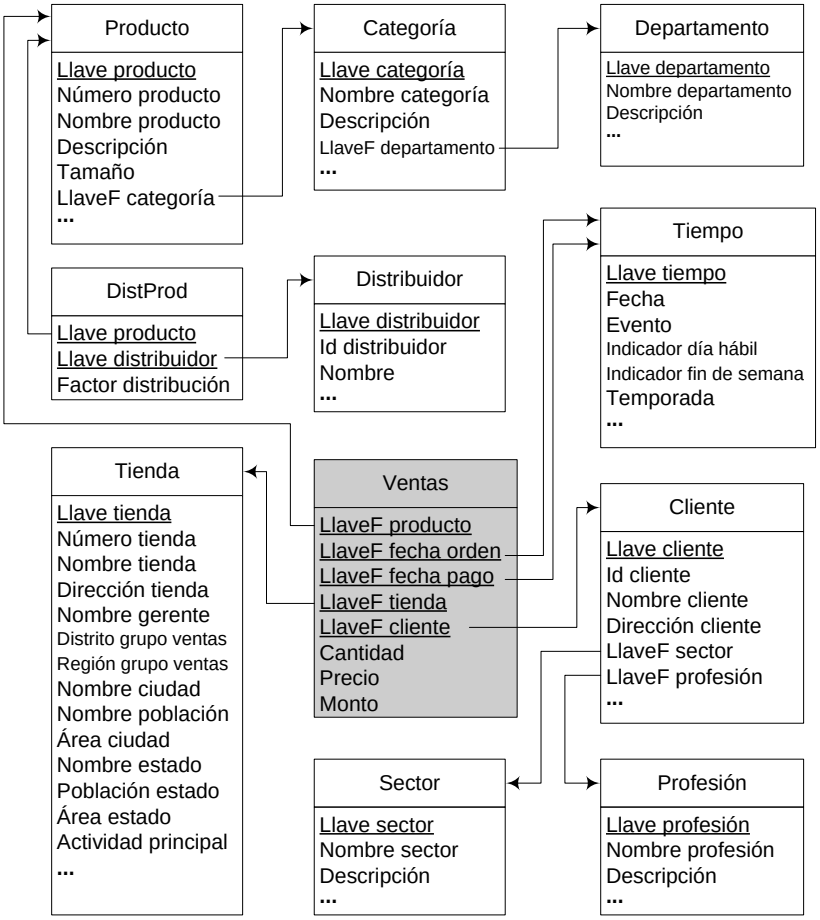


Figura 3.30 Representación relacional del ejemplo en la Figura 3.2

En este breve extracto hemos definido sólo la tabla de hechos Ventas y las tablas para la jerarquía compuesta por los niveles Producto–Categoría–Departamento. Las definiciones para las otras tablas son similares.

La transformación del modelo MultiDim al modelo objeto-relacional se basa en reglas similares a las descritas anteriormente para el mapeo al modelo relacional[14]. La diferencia yace en que, según se explicó en la Sección 2.3.2, el modelo objeto-relacional brinda un tipo referencia (*ref type)*, el cual puede emplearse en lugar de las llaves foráneas descritas en las reglas.

A continuación presentamos el grupo de comandos SQL:2003 que definen un esquema objeto-relacional correspondiente al esquema relacional de la Figura 3.30.

```
create type TipoVenta as (
        RefProducto ref(TipoProducto) scope Producto references are checked,
        RefTiempoOrden ref(TipoTiempo) scope Tiempo references are checked,
        RefTiempoPagoref(TipoTiempo) scope Tiempo references are checked,
        RefTienda ref(TipoTienda) scope Tienda references are checked,
        RefCliente ref(TipoCliente) scope Cliente references are checked,
        Cantidad integer,
        Precio decimal(7,2),
        Monto decimal(7,2) )
        ref is system generated;
create table Ventas of TipoVenta(
        constraint VentasLlaveP primary key (RefProducto, RefTiempoOrden,
                RefTiempoPago, RefTienda, RefCliente),
        RefProducto with options constraint RefProductoNN RefProducto not null,
        /* restricciones no nulas para otros atributos */
        ref is oid system generated );
create type TipoProducto as (
        LlaveProducto integer,
        NúmeroProducto character varying(9),
        NombreProducto character varying(30),
        Descripción text,
        Tamaño integer,
        RefCategoría ref(Tipo Categoría) scope Categoría references are checked,
        ... )
```

[14] Conforme se explicó en la Sección 3.6.2 existe una excepción a esto, ya que la representación relacional y objeto-relacional de las jerarquías no estrictas podría diferir.

```
        ref is system generated;
create table Producto of TipoProducto (
        constraint ProductoLlaveP primary key (ProductoLlave),
        ref is oid system generated );
create type TipoCategoría as (
        LlaveCategoría integer,
        NombreCategoría character varying(20),
        Descripción text,
        RefDepartmento ref(TipoDepartmento) scope Departmento
                references are checked,
        ... )
        ref is system generated;
create table Categoría of TipoCategoría (
        constraint Categoría LlaveP primary key (LlaveCategoría),
        ref is oid system generated );
create type TipoDepartamento as (
        LlaveDepartmento integer,
        NombreDepartamento character varying (30),
        Descripción text,
        ... )
        ref is system generated;
create table Departmento of TipoDepartmento (
        constraint DepartmentoLlaveP primary key (LlaveDepartmento),
        ref is oid system generated );
```

Nótese que hemos implementado el esquema anterior utilizando tablas tipadas. Como se mencionó anteriormente, las tablas tipadas permiten utilizar los identificadores generados por el sistema para enlazar filas entre tablas. Las tablas tipadas también permiten definir métodos para los tipos asociados con las tablas. Sin embargo, puesto que en las secciones siguientes no nos referimos a ningún aspecto relacionado con la manipulación, hemos excluido la especificación de los métodos para los esquemas objeto-relacionales. Nótese también que en las declaraciones de tabla correspondientes especificamos las restricciones asociadas con los tipos (por ejemplo, restricciones de llave primaria). Para efectos de brevedad, en los comandos SQL anteriores no hemos detallado muchas restricciones, en particular no incluimos las restricciones no nulas para

atributos obligatorios. Un ejemplo de este tipo de restricción se encuentra en la tabla Ventas para el atributo RefProducto.

3.6 Representación lógica de las jerarquías

Las reglas generales de mapeo presentadas en la sección anterior no reflejan la semántica específica de todos los tipos de jerarquías descritos en la Sección 3.2. Asimismo, para algunos tipos de jerarquías existen representaciones lógicas alternativas. En esta sección examinamos en detalle la representación lógica de los diferentes tipos de jerarquías. En los próximos párrafos analizamos la representación relacional y la representación objeto-relacional cuando ésta difiere de la relacional.

3.6.1 Jerarquías simples

Jerarquías balanceadas

Cuando aplicamos las reglas de mapeo, explicadas en la Sección 3.5.2, a las jerarquías balanceadas obtendremos las muy conocidas representaciones relacionales de los almacenes de datos:

- *Tablas normalizadas o estructura de copo de nieve*: Cada nivel está representado en una tabla separada que incluye una llave y los atributos descriptivos del nivel. Por ejemplo, al aplicar las Reglas 1 y 2a de la Sección 3.5.2 a la jerarquía Grupos producto en la Figura 3.4, obtenemos una estructura de copo de nieve con las tablas Producto, Categoría y Departamento, según se indica en la Figura 3.31a.
- *Tablas desnormalizadas o planas*: Los atributos llave y descriptivos de todos los niveles que forman una jerarquía se representan en una tabla. Esta estructura puede obtenerse de dos maneras: (1) desnormalizando las tablas que representan varios niveles jerárquicos (por ejemplo, incluir en una tabla todos los atributos de las tablas Producto, Categoría y Departamento que se ilustran en la Figura 3.31a), o (2) mapeando una dimensión que contenga una jerarquía de un nivel que cumpla con la Regla 1 (por ejemplo, la dimensión Tienda en la Figura 3.2 podría representarse tal como se muestra en la Figura 3.31b).

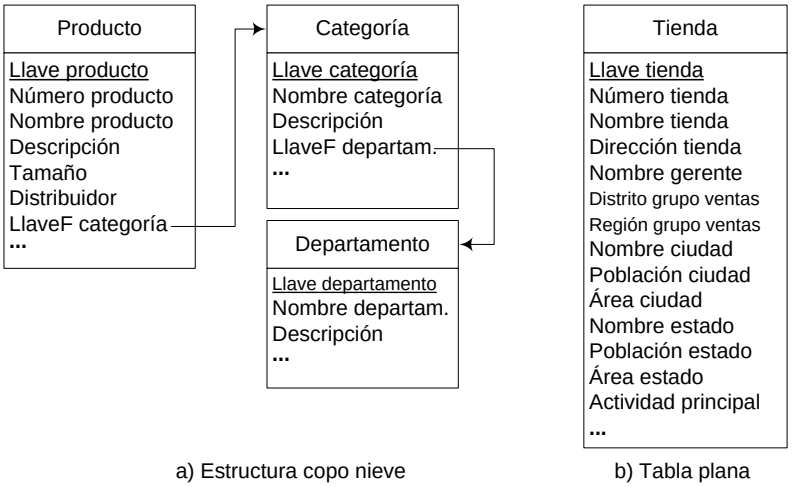


Figura 3.31. Relaciones para una jerarquía balanceada

Las tablas normalizadas se utilizan en los *esquemas de copo de nieve* (véase la Sección 2.7). Este tipo de tabla representa mejor las estructuras jerárquicas, ya que cada nivel puede distinguirse fácilmente y, además, los niveles pueden reutilizarse entre jerarquías diferentes. Adicionalmente, esta representación permite manejar fácilmente la heterogeneidad en los niveles [130], es decir, permite que los diferentes niveles de una jerarquía contengan atributos específicos. Por ejemplo, las tablas Producto, Categoría y Departamento en la Figura 3.31a tienen atributos específicos. Esta estructura de datos acepta que las medidas sean agregadas utilizando, por ejemplo, operadores del *group-by*, *roll-up* y *cube* de SQL (véase la Sección 2.6.3). Además, en algunas aplicaciones, los esquemas de copo de nieve pueden mejorar el desempeño del sistema, pese a que requieren operaciones de mezcla entre las relaciones que representan los niveles [190]. Sin embargo, los esquemas de copo de nieve tienen algunas limitaciones; por ejemplo, si dentro de un nivel hay heterogeneidad —por ejemplo, características diferentes dentro de un grupo de miembros de un nivel— no es posible utilizarlos sin que requieran algún tipo de modificación [130]. Esta situación podría presentarse cuando productos diferentes tienen sus propios atributos específicos, como es el caso de las jerarquías generalizadas (véase la Sección 3.2.1).

Por otra parte, la inclusión de las tablas planas en el esquema que conduce a los *esquemas de estrella*, (véase la Sección 2.7) tiene varias ventajas: el esquema es fácil de entender para la formulación de consultas, puede accederse fácilmente y requiere apenas unas pocas operaciones de mezcla para expresar las consultas debido al alto grado de desnormalización. Asimismo, se han llevado a cabo varios estudios orientados a mejorar el desempeño del sistema durante el procesamiento de consultas de estrella (por ejemplo, [83, 142]). Por último, al igual que con el enfoque normalizado, las agregaciones de las medidas son fáciles de realizar.

Algunas desventajas de la representación del esquema de estrella son las siguientes: No permite que las jerarquías se modelen de manera adecuada, puesto que la estructura jerárquica no es clara y es necesario tener información adicional para representarla [130]. Por ejemplo, para la dimensión Tienda de la Figura 3.31b, no queda claro cuáles atributos pueden utilizarse para las jerarquías. Como también se puede ver en la figura, es difícil asociar claramente los atributos dentro de sus niveles correspondientes. Además, en las jerarquías que tienen muchos niveles, el número de atributos es, por lo menos, tan grande como profunda es la jerarquía, lo cual hace que la estructura jerárquica sea difícil de entender. La desnormalización y, como consecuencia, la redundancia de los datos, también podrían acarrear otros problemas muy conocidos en las bases de datos.

Jerarquías no balanceadas

En vista de que las jerarquías no balanceadas no satisfacen las condiciones de sumarización [164] (véase la Sección 2.6.2), el mapeo presentado en la Sección 3.5.2 podría ocasionar que se excluya del análisis a los miembros de niveles superiores que no tienen miembros hijo. Por ejemplo, dado que en la Figura 3.5a todas las medidas están asociadas con el nivel hoja (es decir, el nivel Cajero) estas medidas se agregarán a niveles más altos únicamente para aquellas agencias que tengan cajeros e, igualmente, sólo para aquellas sucursales que tengan agencias. Para evitar este problema, es posible transformar una jerarquía no balanceada en una balanceada. Esta transformación se lleva a cabo empleando marcadores de posición (indicados con la abreviatura MP en la Figura 3.32) [233] o utilizando valores nulos en los niveles ausentes; luego puede aplicarse el mapeo lógico a las jerarquías balanceadas.

Sin embargo, esta representación tiene las siguientes limitaciones:

1. Una tabla de hechos debe incluir medidas comunes que pertenezcan a diferentes niveles jerárquicos (en nuestro ejemplo, cajeros automáticos, agencias y sucursales).
2. Las medidas comunes utilizadas para todos los niveles tienen diferentes granularidades de datos. Por ejemplo, las medidas para el nivel Cajero y el nivel Agencia para aquellos miembros que no tienen un cajero automático se incluyen en la misma tabla de hechos.
3. Se deben crear y manejar marcadores de posición especiales para realizar la agregación. Por ejemplo, en la Figura 3.32 el mismo valor de medida para la sucursal 2 debe repetirse, a en dos marcadores de posición para los niveles ausentes.
4. El incorporar valores sin sentido exige más espacio de almacenamiento.
5. Se debe desarrollar una interface especial para que los usuarios no vean los marcadores de posición.

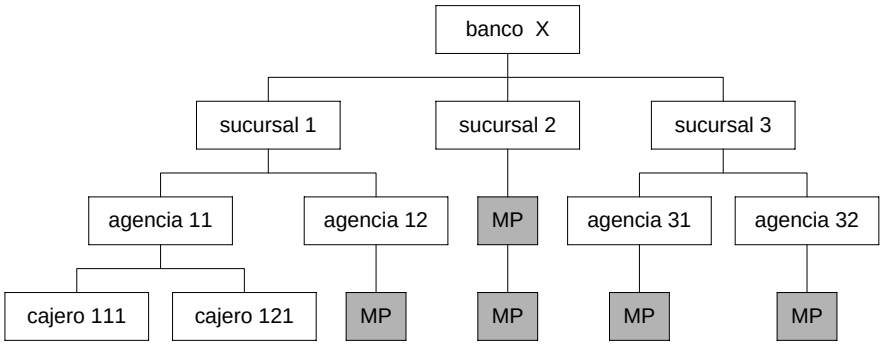


Figura 3.32. Transformación de la jerarquía no balanceada mostrada en la Figura 3.5b en una jerarquía balanceada utilizando marcadores de posición

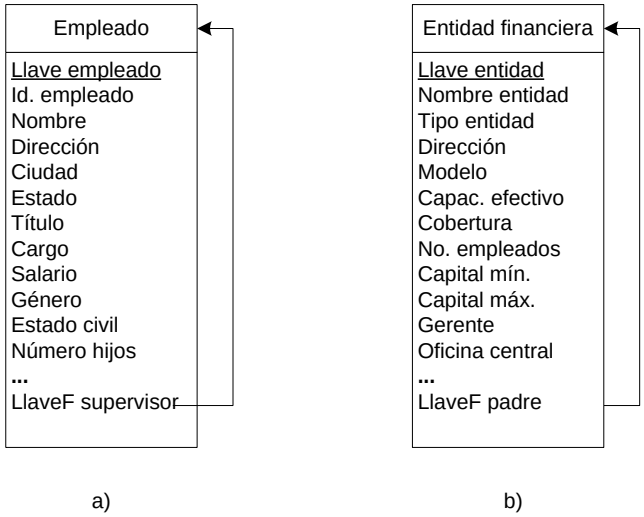


Figura 3.33. Implementación relacional de las jerarquías recursivas mostradas en la Figura 3.6

Como se mencionó en la Sección 3.2.1, las *jerarquías recursivas* son un caso especial de jerarquías no balanceadas. Cuando mapeamos las jerarquías recursivas al modelo relacional obtenemos *tablas padre-hijo* que contienen todos los atributos de un nivel, así como una llave foránea adicional que enlaza a los miembros hijo con su respectivo padre. La Figura 3.33a muestra la tabla que representa la jerarquía recursiva de la Figura 3.6a.

Si bien la tabla padre-hijo nos permite representar la semántica de las jerarquías recursivas, las operaciones que se realizan en ella son más complejas. En particular, es necesario realizar *consultas recursivas* para atravesar los niveles que forman una jerarquía recursiva. Las consultas recursivas se incluyen en el estándar SQL:2003 [194] y pueden manejarse a través de varios sistemas comerciales, tales como SQL Server, Oracle y DB2.

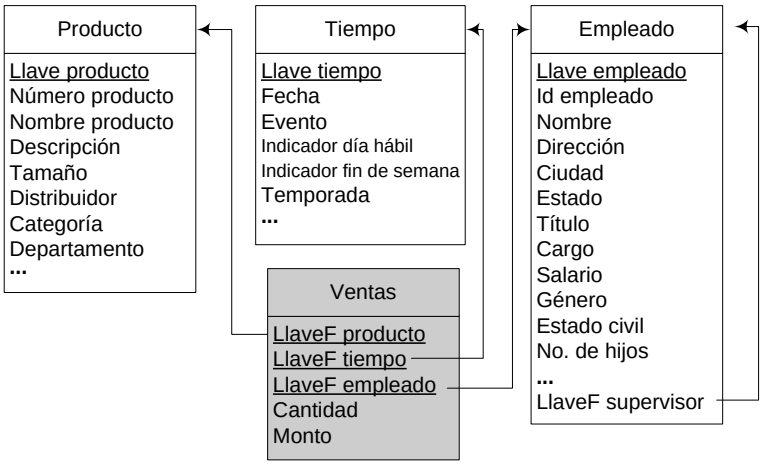


Figura. 3.34. Esquema con una jerarquía recursiva en la dimensión Empleado

Consideremos el ejemplo presentado en la Figura 3.34 de un almacén de datos creado para analizar las ventas efectuadas por los empleados por medio de mercadeo directo, es decir, cuando los bienes se venden directamente al cliente. Supongamos ahora que el usuario está interesado en encontrar, para cada supervisor, el monto total de las ventas generadas por sus subordinados durante el año 2006. Esto requiere una consulta recursiva en SQL, según se muestra a continuación.

```
with Supervision(Llave supervisor, Llave subordinado) as
(       select LlaveF supervisor, Llave empleado
        from Empleado
        where LlaveF supervisor is not null
union all
        select S.Llave supervisor , E.Llave empleado
        from Supervision S, Empleado E
        where S.Llave subordinado = E.LlaveF supervisor
)
select Llave supervisor, sum(S1.Monto)
from Ventas S, Supervision U, Tiempo T
where S.LlaveF empleado=U.Llave subordinado
and S.LlaveF tiempo=T.Llave tiempo
and T.Fecha >= '1/1/2006' and T.Fecha =< '31/12/2006'
group by Llave supervisor
```

En esta consulta, la cláusula with define a la tabla temporal Supervision que calcula recursivamente el número de todos los subordinados directos e indirectos de un empleado. Por ejemplo, si E1 supervisa a E2 y E2 supervisa a E3, la tabla Supervision contendrá además una línea que explique que E1 supervisa a E3. Luego, la declaración select une las tablas Supervision, Ventas y Tiempo y agrupa el resultado usando Llave supervisor para sumar todas las ventas de los subordinados.

Entidad financiera											
Llave entidad	Nombre entidad	Tipo entidad	Direc-ción	Modelo	Capac. efectivo	Cober-tura	No. emp.	Capital mín.	Capital máx.	...	LlaveF padre
1	cajero111	Cajero	…	T1	100000	nulo	nulo	nulo	nulo	…	3
2	cajero112	Cajero	…	T2	150000	nulo	nulo	nulo	nulo	…	3
3	agencia11	Agencia	…	nulo	nulo	150	12	nulo	nulo	…	5
4	agencia12	Agencia	…	nulo	nulo	135	9	nulo	nulo	…	5
5	sucursal1	Sucursal	…	nulo	nulo	nulo	nulo	10000	99000	…	10
6	sucursal2	Sucursal	…	nulo	nulo	nulo	nulo	1500	55000	…	10
7	agencia31	Agencia	…	nulo	nulo	230	15	nulo	nulo	…	9
8	agencia32	Agencia	…	nulo	nulo	185	12	nulo	nulo	…	9
9	sucursal3	Sucursal	…	nulo	nulo	nulo	nulo	7500	98500	…	10
10	bancoX	Banco	…	nulo	nulo	nulo	nulo	nulo	nulo	…	nulo

Figura. 3.35. Esquema relacional padre-hijo de una jerarquía no balanceada con instancias de la Figura 3.5b

Una solución para implementar jerarquías no balanceadas es transformarlas en jerarquías recursivas que luego se convierten, a nivel lógico, en *tablas padre-hijo*. Por ejemplo, la transformación de la jerarquía recursiva de la Figura 3.6b producirá la tabla mostrada en la Figura 3.33b. No obstante, el problema de estas tablas es que semánticamente representan niveles diferentes, cada uno con atributos particulares. Como puede apreciarse en la Figura 3.35, si una tabla padre-hijo contiene los atributos de todos los niveles jerárquicos, el miembro de un nivel contendrá valores nulos para todos los atributos que pertenecen a los otros niveles.

Por esta razón, se deben implementar *restricciones de integridad* con el fin de garantizar que un miembro tenga valores para los atributos de su nivel. Por ejemplo, la siguiente restricción de verificación (*check*) establece que los miembros del nivel Cajero deben tener valores para los atributos Dirección, Modelo y Capac. efectivo y valores nulos para los demás atributos:

```
alter table Entidad financiera
        add constraint Atributos cajero
        check ( (TipoEntidad != 'Cajero') or ( Dirección is not null and
        Modelo is not null and Capac. efectivo is not null and
        Cobertura is null and No. emp. is null and ... ) )
```

Nótese que en la restricción anterior la implicación lógica (TipoEntidad= 'Cajero') ⇒ (. . .) está codificada como (TipoEntidad != 'Cajero') ∨ (. . .). Es necesario contar con restricciones de verificación (*check*) similares para los otros niveles de la jerarquía, es decir, para los niveles Agencia, Sucursal y Banco.

Asimismo, es necesario contar con disparadores con el fin de garantizar que los enlaces entre niveles estén correctos. Por ejemplo, el siguiente disparador, implementado en Microsoft SQL

Server, obliga a que se cumpla la restricción de que los miembros de Cajero deben estar relacionados con los miembros de Agencia:

```
create trigger Cajero_Agencia on Entidad financiera
after insert, update as
if exists ( select * from Inserted I, Entidad financieraF
        where I.LlaveF padre = F.Llave entidad and
        I.Tipo entidad = 'Cajero' and F.Tipo entidad != 'Agencia' )
begin
        raiserror 13000 'El miembro Cajero debe estar relacionado con el miembro Agencia'
end
```

Jerarquías generalizadas

Se han propuesto varias soluciones para la representación lógica de las jerarquías generalizadas: (1) representar la heterogeneidad dentro de los niveles creando tablas separadas para cada nivel de la jerarquía y conservando la información acerca de cuáles tablas componen las diferentes rutas [130]; (2) emplear una representación plana con valores nulos para los atributos que no describen miembros específicos [163] (por ejemplo, las compañías tendrán valores nulos en aquellos atributos que correspondan a personas); (3) utilizar esquemas de estrella separados, es decir, tablas de hechos y de dimensiones separadas para cada ruta de la jerarquía [16, 95, 147]; y (4) crear una tabla para los niveles comunes y otra para los específicos [16, 275].

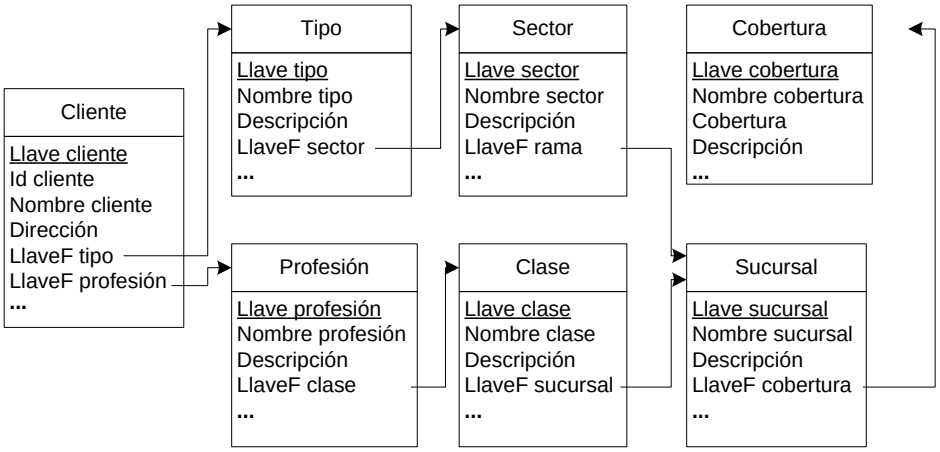


Figura. 3.36. Relaciones para la jerarquía generalizada de la Figura 3.8

Una desventaja de los primeros tres enfoques anteriores es que no es posible distinguir ni manejar con facilidad los niveles comunes de la jerarquía. Además, la inclusión de los valores nulos requiere que establezcamos restricciones adicionales para garantizar que las consultas estén correctas (por ejemplo, no agrupar Tipo con Profesión en la Figura 3.8). Si bien estas dificultades no se presentan para la solución (4), en ese caso debe crearse un atributo adicional en la tabla que represente a los niveles comunes de la jerarquía para especificar el nombre de la

tabla para los niveles específicos. En consecuencia, las consultas que manejan metadatos y datos deben acceder a las tablas de los niveles específicos, lo cual no es tarea fácil en SQL. El mapeo tradicional de generalización/especialización a tablas relacionales (por ejemplo, la Regla 7 en la Sección 2.3.1) también genera problemas debido a que se incluyen valores nulos y se pierde la estructura jerárquica.

Si aplicamos el mapeo descrito en la Sección 3.5.2 a la jerarquía generalizada de la Figura 3.8 obtenemos como resultado las relaciones que se muestran en la Figura 3.36. Aunque este mapeo representa claramente la estructura jerárquica, no nos permite atravesar sólo los niveles comunes de la jerarquía. Para asegurarnos de que haya heterogeneidad dentro de los niveles y que sea posible atravesar los niveles comunes o específicos, proponemos además la siguiente regla de mapeo:

Regla 4: Una tabla que corresponda al nivel de separación en una jerarquía generalizada podría tener un atributo adicional, que sería una llave foránea del próximo nivel de unión, siempre y cuando exista este nivel. La tabla podría también incluir un atributo de diferenciación que indique la ruta específica de agregación de cada miembro.

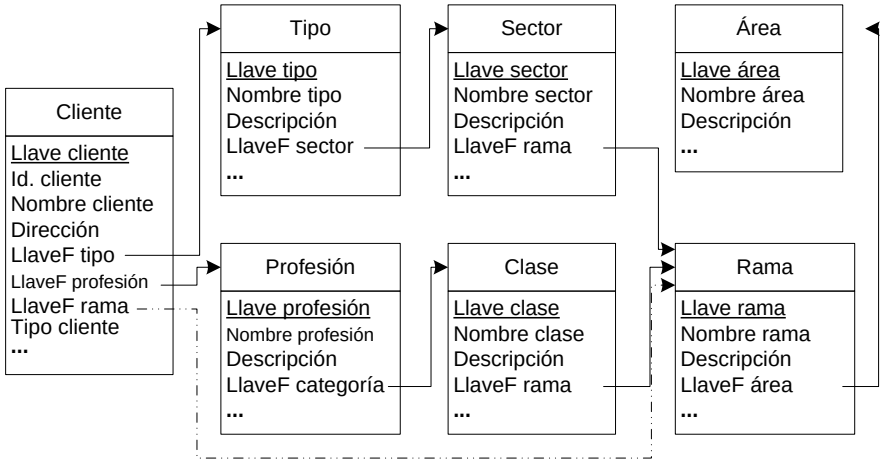


Figura. 3.37. Representación relacional mejorada para la jerarquía generalizada de la Figura 3.8

En la Figura 3.37 se brinda un ejemplo de las relaciones para la jerarquía de la Figura 3.8. La tabla Cliente incluye dos tipos de llaves foráneas: una que indica el próximo nivel jerárquico especializado (por ejemplo, LlaveF profesión), que se obtiene al aplicar las Reglas 1 y 2a de la Sección 3.5.2, y otra correspondiente al próximo nivel de unión (por ejemplo, LlaveF sucursal), el cual se obtiene aplicando la Regla 4. El atributo de diferenciación Tipo de cliente, con los valores Persona y Compañía, indican la ruta específica de agregación de los miembros para facilitar las agregaciones. Por último, debemos especificar las restricciones de integridad con el fin de asegurarnos de que sólo una de las llaves foráneas para los niveles especializados tenga asignado un valor, de acuerdo con el valor del atributo de diferenciación:

```
alter table Cliente
        add constraint VerTipoCliente
        check ( TipoCliente in ('Persona', 'Compañía') )
alter table Cliente
```

```
        add constraint ClientePersonalLlaveF
        check ( (TipoCliente != 'Persona') or
        ( LlaveF Profesión is not null and LlaveF tipo is null ) )
alter table Cliente
        add constraint ClienteCompañíaLlaveF
        check ( (TipoCliente != 'Compañía') or
        ( LlaveF Profesión is null and LlaveFTipo is not null ) )
```

La estructura de la Figura 3.37 nos permite seleccionar rutas alternativas para el análisis. Una posibilidad es utilizar rutas que incluyan niveles específicos, por ejemplo, Profesión o Tipo. Otra posibilidad es acceder a los niveles que son comunes a todos los miembros, en otras palabras, ignorar los niveles que están entre los niveles de unión y de separación; por ejemplo, podríamos analizar a todos los clientes, sin importar su tipo, utilizando la jerarquía Cliente-Sucursal-Cobertura.

Tal como sucede con la estructura de copo de nieve, esta estructura tiene la desventaja de que requiere que se lleven a cabo operaciones de mezcla entre varias tablas. No obstante, una ventaja importante es que ofrece un mayor número de posibilidades de análisis al tomar en consideración las características específicas y generales de los miembros de las jerarquías.

Dado que la *jerarquía no cubierta* es un caso especial de una jerarquía generalizada, también podemos aplicar el mapeo propuesto anteriormente. Sin embargo, puesto que en una jerarquía no cubierta existe una ruta única en la que se omiten sólo algunos niveles, la solución habitual (por ejemplo [101, 233]) es transformar la jerarquía incorporando marcadores de posición en los niveles intermedios ausentes. De esta manera, podemos convertir una jerarquía no cubierta en una balanceada y utilizar una estructura de estrella o de copo de nieve para hacer la representación lógica.

3.6.2 Jerarquías no estrictas

El mapeo tradicional de las jerarquías no estrictas al modelo relacional, según se especificó en la Sección 3.5.2, crea relaciones para representar los niveles y una relación adicional para representar las relaciones muchos a muchos entre ellos. En la Figura 3.38 se muestra un ejemplo para el caso de la jerarquía de la Figura 3.11a. La tabla que representa la relación muchos a muchos (SecciónEmpl en la figura) normalmente se conoce como *tabla puente* [147].

Vale notar que si la relación padre-hijo contiene un factor de distribución, la tabla puente incluirá un atributo adicional para almacenar los valores requeridos para la distribución de medidas.[15] Por ejemplo, para un empleado X, la tabla SecciónEmpl. podría incluir tres filas: (1) empleado X, sección 1, 30%; (2) empleado X, sección 2, 50%; (3) empleado X, sección 3, 20%. Para poder agregar las medidas correctamente se debe implementar un procedimiento especial de agregación que utiliza este factor de distribución.

[15] En [147] se le llama factor de ponderación.

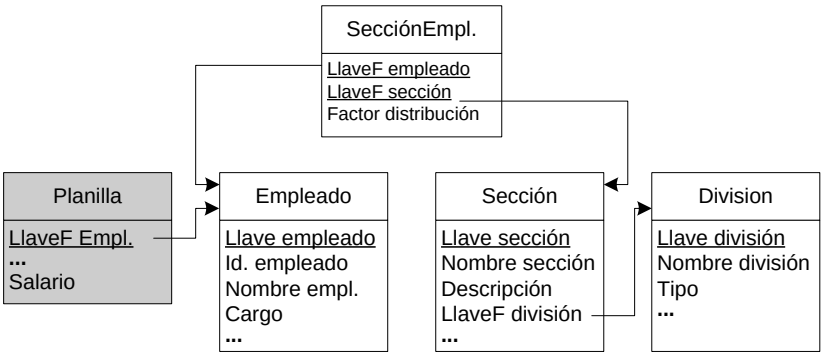


Figura. 3.38. Tablas relacionales para la jerarquía no estricta de la Figura 3.11

En la Sección 3.2.2 vimos que otra solución es transformar una jerarquía no estricta en una estricta incluyendo una dimensión adicional en la relación de hechos, según se ilustra en la Figura 3.14, y luego aplicar el correspondiente mapeo para una jerarquía estricta. La elección entre dos soluciones podría depender de varios factores, tales como los mencionados en la Tabla 3.1. Aún así, existe otra opción que consiste en convertir la relación muchos a muchos en una relación uno a muchos definiendo una relación primaria [203], es decir, convirtiendo la jerarquía no estricta en una estricta, a la cual se le aplicaría el correspondiente mapeo para una jerarquía simple.

Existen dos posibilidades para representar jerarquías no estrictas en el modelo objeto-relacional. Podemos utilizar una solución similar a la tabla puente que se mencionó anteriormente, con la diferencia de que en lugar de utilizar llaves foráneas utilizamos referencias. La otra solución es ampliar la tabla que representa un nivel hijo por medio de un atributo multivaluado adicional (de tipo *array* o *multiset*) que incluya referencias a los miembros padre. Si el esquema contiene un factor de distribución, el atributo multivaluado adicional está compuesto por dos atributos, tal como se ilustra en la Figura 3.39: uno que hace referencia al miembro padre y otro que especifica el valor del factor de distribución para este miembro padre específico.

Cuadro 3.1
Comparación de implementaciones de jerarquías no estrictas utilizando una tabla puente y una dimensión adicional

Factor	Tabla puente	Dimensión adicional
Tamaño de tabla	La tabla de hechos es más pequeña.	La tabla de hechos es más grande si los miembros hijo están relacionados con muchos miembros padre. La llave foránea adicional en la tabla de hechos también incrementará el espacio requerido.
Estructura adicional	La información adicional acerca de la relación padre-hijo y el factor de distribución (si existe) debe almacenarse por separado.	Las medidas se incluyen únicamente en la tabla de hechos.
Desempeño	Es necesario realizar un número adicional de operaciones de mezcla, cálculos y tareas de programación para poder agregar las medidas correctamente.	Las medidas en la tabla de hechos están listas para realizar la agregación a lo largo de la jerarquía.
Tipo de aplicación	Útil para aplicaciones que tienen únicamente unas cuantas jerarquías no estrictas y otras jerarquías estrictas que se refieran a las mismas medidas.	La implementación debe evaluarse tomando en cuenta las ventajas y desventajas comparativas entre tiempo y espacio.
Cambios con el tiempo	Útil cuando la información sobre la distribución de medidas no cambia con el tiempo.	Puede representar con facilidad cambios en la distribución de medidas que ocurren con el tiempo.
Implementación OLAP actual	Las herramientas actuales pueden manejar este tipo de jerarquía hasta cierto punto.	Las herramientas actuales pueden manejar este tipo de jerarquía, pero en algunos casos podrían surgir problemas de conteo doble.

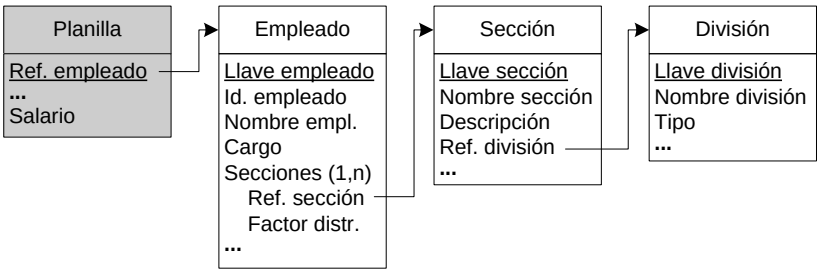
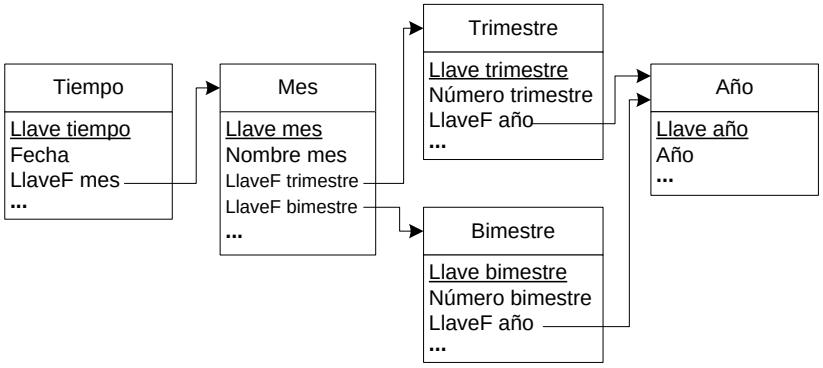


Figura. 3.39. Representación objeto-relacional de la jerarquía no estricta de la Figura 3.13

Si bien las jerarquías no estrictas son muy comunes en situaciones de la vida real, no se ha investigado mucho sobre ellas. El buscar soluciones que permitan optimizar el manejo de estas jerarquías reviste suma importancia [104]. En esta sección, hemos analizado solamente el caso de las jerarquías balanceadas no estrictas. Cuando tenemos una jerarquía no estricta de uno de los otros tipos presentados anteriormente, debemos utilizar una combinación del mapeo propuesto para la jerarquía específica y del mapeo para una jerarquía no estricta.



3.6.3 Jerarquías alternativas

Para las jerarquías alternativas, podemos aplicar el mapeo tradicional a las tablas relacionales. En la Figura 3.40 mostramos el mapeo realizado para el esquema conceptual de la Figura 3.16. Nótese que, a pesar de que las jerarquías generalizadas y alternativas pueden distinguirse fácilmente a nivel conceptual (véanse las Figuras 3.8a y 3.16), esta diferenciación no es factible a nivel lógico (compárense las Figuras 3.36 y 3.40).

Figura. 3.40. Relaciones para las jerarquías alternativas de la Figura 3.16

3.6.4 Jerarquías paralelas

Dado que las jerarquías paralelas están compuestas ya sea de jerarquías alternativas o bien de jerarquías simples, su mapeo lógico consiste en combinar los mapeos para tipos específicos de jerarquías. Por ejemplo, la Figura 3.41 muestra el resultado de aplicar este mapeo al modelo relacional para el esquema mostrado en la Figura 3.18.

Nótese que los niveles compartidos en las jerarquías paralelas dependientes están representados en una tabla. Puesto que este nivel juega diferentes roles en cada jerarquía, durante la implementación es necesario crear vistas para evitar la repetición innecesaria de datos y facilitar las consultas. Vale notar también que ambas jerarquías alternativas y paralelas dependientes pueden distinguirse fácilmente a nivel conceptual (Figuras 3.16 y 3.18). No obstante, sus representaciones a nivel lógico (Figuras 3.40 y 3.41) parecen ser similares a pesar de tener varias características que las diferencian, según se explicó en la Sección 3.2.4.

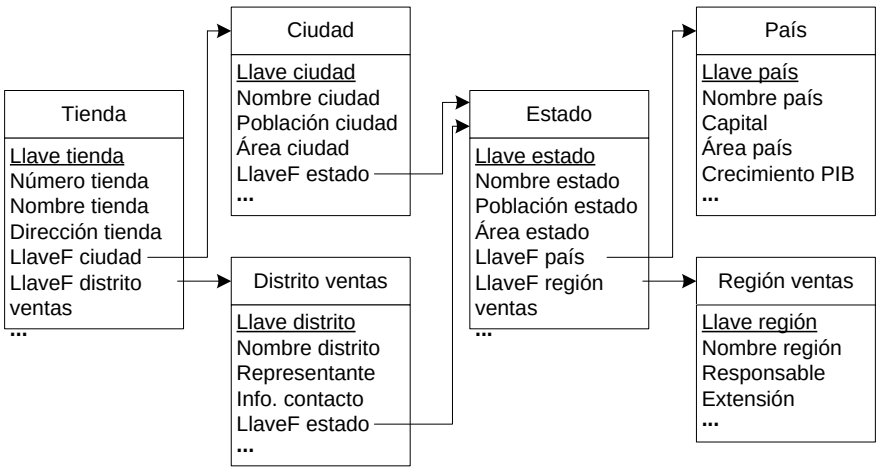


Figura. 3.41. Esquema relacional de un conjunto de jerarquías paralelas dependientes compuestas por dos jerarquías balanceadas

3.7 Implementación de jerarquías

Esta sección describe la implementación de los diferentes tipos de jerarquías analizadas en este capítulo en dos herramientas comerciales, Microsoft Analysis Services 2005 y Oracle OLAP 10g.

3.7.1 Jerarquías en Analysis Services 2005

Jerarquías balanceadas

Como ya se mencionó en la Sección 2.10, tanto la representación del esquema de copo de nieve como el de estrella pueden manejarse con Analysis Services [101]. Recordemos que lo que nosotros llamamos jerarquías en Analysis Services se conocen como *jerarquías definidas por el usuario* o *jerarquías de niveles múltiples*. Estas últimas pueden definirse por medio de todos los atributos en una dimensión; dichos atributos pueden almacenarse en una sola tabla o en varias tablas de un esquema de copo de nieve.

Jerarquías no balanceadas

Analysis Services propone la implementación de jerarquías no balanceadas utilizando jerarquías padre-hijo. No obstante, esto implica que las medidas deben ser iguales para todos los niveles jerárquicos.

En una jerarquía padre-hijo, los miembros que no son miembros hoja podrían también tener medidas asociadas en una tabla de hechos, además de las medidas agregadas desde sus miembros hijo. Para este tipo de miembro se crean miembros hijo especiales generados por el sistema que contienen los datos subyacentes de la tabla de hechos. Estos miembros hijo se conocen como *miembros de datos* y siempre se incluyen como miembros hijo para propósitos de agregación. Sin embargo, el comportamiento normal de agregación puede reemplazarse utilizando una fórmula *personalizada de navegación ascendente*.

La propiedad MemberswithData se utiliza para controlar la visibilidad de los miembros de datos, los cuales se muestran cuando esta propiedad se fija con la instrucción NonLeafDataVisible en el atributo padre y se esconden con NonLeafDataHidden. La propiedad MembersWithDataCaption del atributo padre nos permite definir una plantilla para generar nombres que se asignan a los miembros de datos.

Jerarquías generalizadas

Analysis Services no maneja jerarquías generalizadas. Si los miembros difieren en los atributos y en la estructura jerárquica normalmente lo que procede es definir una jerarquía para los niveles comunes y otra para cada una de las rutas exclusivas que contienen los niveles específicos.

Por otra parte, Analysis Services maneja *jerarquías no cubiertas* llamadas *jerarquías desiguales (ragged)*. Una jerarquía desigual se define utilizando todos sus niveles (es decir, la ruta más larga). Los miembros que no incluyen un nivel determinado en la jerarquía podrían contener valores nulos o cadenas de caracteres vacías, o contener el mismo valor de su miembro padre para funcionar como marcador de posición. La propiedad de dimensión HideMemberIf se utiliza

para desplegar estos miembros de manera apropiada. A continuación se enumeran los posibles valores para la propiedad HideMemberIf y sus respectivos comportamientos:

- Never: Los miembros de un nivel nunca se ocultan.
- OnlyChildWithNoName: El miembro de un nivel se oculta cuando es el único hijo de su padre y su nombre es nulo o es una cadena de caracteres vacía.
- OnlyChildWithParentName: El miembro de un nivel se oculta cuando es el único hijo de su padre y su nombre es el mismo que el de su padre.
- NoName: El miembro de un nivel se oculta cuando su nombre está vacío.
- ParentName: El miembro de un nivel se oculta cuando su nombre es idéntico al de su padre.

Jerarquías no estrictas

Analysis Services 2005 nos permite definir relaciones muchos a muchos entre una dimensión y una relación de hechos. Esto se logra utilizando una tabla puente como la de la Figura 3.38. En el caso de esta tabla (usando la terminología empleada por Analysis Services) la dimensión Sección tiene una relación muchos a muchos con el grupo de medidas Planilla, por medio de la dimensión intermedia Empleado y el grupo de medidas SecciónEmpl.

Jerarquías alternativas

Varias jerarquías pueden estar conectadas a una dimensión y tener niveles compartidos. Por ejemplo, es posible definir dos jerarquías en la dimensión Tiempo de la Figura 3.16: una compuesta por Tiempo-Mes-Trimestre-Año y otra por Tiempo-Mes-Bimestre-Año.

Sin embargo, no existe una noción de jerarquías alternativas puesto que las jerarquías no pueden definirse como exclusivas. Por ejemplo, si combinamos las dos jerarquías de la Figura 3.16 obtendremos intersecciones sin sentido como T2-2001 y B1-2001. Aunque Analysis Services manipule las medidas adecuadamente otorgando a dichas intersecciones un despliegue vacío, la herramienta debería cambiar automáticamente entre las jerarquías exclusivas que la componen. Por ejemplo, si un usuario está analizando medidas con la jerarquía Tiempo-Mes-Trimestre-Año y solicita la jerarquía Tiempo-Mes-Bimestre-Año, la primera jerarquía debería excluirse del análisis actual.

Jerarquías paralelas

Analysis Services permite manejar jerarquías paralelas, ya sean dependientes o independientes. Los niveles pueden ser compartidos entre las varias jerarquías que los componen.

3.7.2 Jerarquías en Oracle OLAP 10g

Los pasos generales para definir jerarquías en Oracle OLAP 10g [214] son, primero, crear una dimensión; luego, los niveles; seguidamente, las jerarquías indicando los niveles que las componen; y, por último, realizar el mapeo de los niveles a los datos fuente.

Como se mencionó en la Sección 2.11, Oracle OLAP brinda dos tipos de jerarquías: las basadas en el nivel y las basadas en el valor. Las primeras requieren que se definan los niveles que las componen, lo cual corresponde usualmente a las columnas en una tabla relacional (en forma

plana o normalizada). Por otra parte, las jerarquías basadas en el valor se forman tomando en cuenta las relaciones entre miembros que se definen en una tabla padre-hijo.

Con el fin de garantizar que los miembros que forman una dimensión sean únicos, Oracle OLAP permite a los diseñadores utilizar llaves naturales o sustitutas. Las llaves naturales se leen desde la fuente relacional sin modificación alguna. Las llaves sustitutas se crean antes de que se carguen los miembros al área analítica de trabajo y consisten en el prefijo del nombre del nivel sumado a cada miembro. Puesto que los niveles no se definen en las jerarquías basadas en el valor, Oracle OLAP impone la restricción de que únicamente las llaves naturales, y no las sustitutas, pueden usarse con las jerarquías basadas en el valor.

Jerarquías balanceadas

Oracle OLAP se refiere a las jerarquías balanceadas como jerarquías *normales*. Se definen como jerarquías basadas en el nivel, es decir, la estructura jerárquica se forma en términos de niveles y relaciones entre ellas. Oracle OLAP permite realizar el mapeo de este tipo de jerarquías a la representación de los esquemas de estrella y de copo de nieve, es decir, a columnas almacenadas en una sola tabla o en varias tablas.

Jerarquías no balanceadas

En Oracle OLAP las jerarquías no balanceadas se conocen como *jerarquías desiguales (ragged)*. Oracle OLAP brinda dos soluciones para implementarlas, dependiendo de si los niveles pueden distinguirse o no. Una opción es definir jerarquías no balanceadas como jerarquías basadas en el nivel, lo cual requiere incluir valores nulos en las columnas de la tabla fuente que corresponden a los miembros hijo ausentes. Otra opción es definir las jerarquías no balanceadas como jerarquías basadas en el valor, cuyos datos provienen de las tablas padre-hijo. Nótese que las *jerarquías recursivas* pueden implementarse sólo como jerarquías basadas en el valor.

Jerarquías generalizadas

Actualmente no es posible manejar jerarquías generalizadas en Oracle OLAP. Pueden definirse ya sea como una jerarquía que incluye los niveles comunes o como cubos separados, cada uno con una dimensión formada por una jerarquía específica a un grupo de miembros.

Por otro lado, Oracle OLAP maneja *jerarquías no cubiertas* que denomina *jerarquías que omiten niveles*, las cuales también pueden definirse ya sea como jerarquías basadas en el nivel o jerarquías basadas en el valor. Para definir jerarquías no cubiertas como jerarquías basadas en el nivel es necesario que la tabla fuente contenga valores nulos en las columnas que corresponden a los niveles ausentes. Cuando una jerarquía no cubierta se define como jerarquía basada en el valor, las relaciones entre miembros de la jerarquía se representan mediante una tabla padre-hijo.

Jerarquías no estrictas

Oracle OLAP no permite manejar jerarquías no estrictas.

Jerarquías Alternativas

Oracle OLAP no distingue entre jerarquías alternativas y paralelas. Por lo tanto, aunque los desarrolladores definan varias jerarquías dentro de una dimensión, es responsabilidad de los usuarios saber que el combinar las jerarquías que la componen no producirá ningún resultado y, por lo tanto, no debe hacerse.

Jerarquías paralelas

Puesto que Oracle OLAP permite manejar la definición de varias jerarquías dentro de una misma dimensión, las jerarquías paralelas independientes pueden implementarse definiendo cada jerarquía por separado e indicando los niveles que la componen.

Por otra parte, para las jerarquías paralelas dependientes, cada nivel compartido debe definirse por separado para cada jerarquía que lo incluye. Además, para garantizar que cada miembro que pertenece a la dimensión sea único, los miembros de niveles repetidos deben tener diferentes llaves sustitutas asignadas a ellos.

3.8 Trabajos realizados sobre el tema

Desde hace varias décadas se han venido reconociendo las ventajas que ofrece el modelado conceptual para el diseño de bases de datos y se han examinado en muchas publicaciones (por ejemplo, [30, 46]. Sin embargo, el análisis presentado en [262] muestra el escaso interés de la comunidad de investigación en el modelado multidimensional conceptual. En [251, 299] se ofrece una descripción detallada de los modelos multidimensionales. Algunas propuestas brindan representaciones gráficas basadas en el modelo ER ([266, 302]), o en el Lenguaje Unificado de Modelado (UML) ([5, 301]), o bien proponen nuevas notaciones ([83, 110, 305]), mientras que otras propuestas no se refieren a la representación gráfica ([107, 233, 245]). Otras publicaciones relacionadas con los sistemas de almacenes de datos y OLAP se refieren al problema de sumarización (por ejemplo, [105, 106, 107, 164]), las formas normales multidimensionales (por ejemplo, [161, 163]) o a las consideraciones acerca de la implementación (por ejemplo, [58, 142]), entre otros temas.

Las publicaciones que se revisan en esta sección se dividen en tres categorías: Primero, las que describen varios tipos de jerarquías; segundo, aquellas referidas a las representaciones lógicas de las jerarquías; y, por último, las que ofrecen soluciones comerciales para la implementación de las jerarquías y que mencionamos aquí brevemente. A los efectos de una mejor comparación, en la siguiente descripción utilizamos nuestra propia terminología para los diferentes tipos de jerarquías.

Existe una amplia variedad de tipos de jerarquías que manejan los modelos multidimensionales actuales. Algunos modelos (por ejemplo, [83, 266, 299]) permiten manejar únicamente jerarquías simples. Esta situación es una deficiencia de los modelos existentes para los almacenes de datos [104]. La Tabla 3.2 compara los modelos multidimensionales que, hasta donde sabemos, manejan jerarquías. Para esta comparación hemos utilizado tres símbolos: — cuando no existe referencia al tipo específico de jerarquía, ± cuando se presenta sólo una descripción y/o definición de la jerarquía y √ cuando se ofrece una descripción y una representación gráfica. Si se propone un nombre diferente para una jerarquía, éste se incluye en la tabla.

Como puede verse en la Tabla 3.2, los modelos multidimensionales existentes no toman en cuenta todos los tipos de jerarquías descritos en este libro. Varios modelos brindan solamente una descripción y/o definición de algunas de estas jerarquías, sin dar una representación gráfica (por ejemplo, [105, 130, 208, 209, 233, 245]). Además, para el mismo tipo de jerarquía se emplean diferentes nombres (por ejemplo en el caso de jerarquías no balanceadas, generalizadas y no estrictas). Si bien algunas propuestas se refieren a las jerarquías generalizadas (por ejemplo, [170]), algunas incluyen el supertipo y los subtipos como niveles jerárquicos sin considerar los atributos que forman una jerarquía, como sí lo hace nuestro modelo. Todos los modelos

propuestos manejan jerarquías estrictas explícita o implícitamente. Ninguno de los modelos toma en cuenta la posibilidad de que puedan existir varios criterios de análisis diferentes; en consecuencia, no es posible distinguir las jerarquías alternativas ni las paralelas. Como ya lo explicamos en la Sección 3.2.4, es importante hacer esta diferenciación para no asumir que cada ruta en las jerarquías paralelas debe conducir a los mismos miembros de un nivel compartido [160]. Muy pocos modelos proponen una representación gráfica para diferentes jerarquías que faciliten su distinción a nivel de esquema y de instancias (por ejemplo, [170, 266, 302]).

El Grupo de Gestión de Objetos (OMG) ha propuesto el modelo Común de Almacenes de Datos (CWM) como estándar para representar sistemas de almacenes de datos y OLAP. Este modelo brinda un marco de referencia para representar metadatos de fuentes de datos, de datos de destino, transformaciones y análisis, así como de procesos y operaciones para crear y gestionar los datos de los almacenes. Este modelo está representado como una estructura en capas que contiene varios submodelos. Uno de estos submodelos, la capa de recursos, define modelos que pueden utilizarse para representar información en los almacenes de datos, incluido el modelo relacional. Adicionalmente, la capa de análisis presenta un metamodelo para OLAP, que incluye los conceptos de dimensión y jerarquía. En el CWM es posible representar todos los tipos de jerarquías detalladas en la Tabla 3.2.

Tabla 3.2
Comparación de los modelos conceptuales que manejan varios tipos de jerarquías

Modelo	Balanceada	No balanceada	Generalizada	No cubierta	No estricta	Alternativa o paralela dependiente
[233]	± Onto	± Non-onto	—	± No cubierta	± No estricta	± Múltiple
[110]	√ Simple	—	√ Opcional múltiple	—	—	√ Alternativa
[302]	√	√	—	—	√ No estricta	√ Múltiple
[105, 107]	± Estrictamente homogénea	± Homogénea	± Heterogénea	± Heterogénea	—	± Heterogénea
[245]	± Clasificación total	± Clasificación parcial	± Múltiple	± Múltiple	—	± Multiplicidad múltiple
[130]	±	± No balanceada	± Heterogénea	± Heterogénea	—	±
[16]	—	—	±	—	—	—
[5]	√	± Non-onto	—	±	±	√
[170]	√	±	± Especialización	√	√	√ Alternativa
[305]	√	—	—	√	±	—
[83, 266, 299]	√	—	—	—	—	√
[208]	± Balanceada	± No balanceada	–	± Desigual	—	—
[209]	± Basada en el nivel	± Basada en el valor	± Basada en el valor	± Basada en el valor	—	—

Las herramientas OLAP comerciales que existen en la actualidad no permiten realizar el modelado conceptual de las jerarquías. Generalmente, brindan una representación a nivel lógico limitada a los esquemas de estrella y copo de nieve. En las secciones anteriores nos hemos referido a las publicaciones que utilizan el enfoque relacional para representar varios tipos de jerarquías, donde las jerarquías complejas algunas veces se transforman en simples. Por ejemplo, examinamos los enfoques adoptados por Pedersen et al. [233] para transformar jerarquías no balanceadas, no cubiertas y no estrictas y las soluciones propuestas por Kimball y Ross [147] para manejar jerarquías balanceadas, generalizadas y no estrictas. También mencionamos las ventajas y desventajas de las soluciones propuestas. A pesar de que algunas de estas soluciones (por ejemplo, [130]) brindan un mapeo correcto que capta la semántica de la jerarquía, también generan un número significativo de relaciones y requieren extensiones SQL para su manejo. Por otro lado, Song et al. [283] han propuesto varias soluciones lógicas al problema de manejar relaciones muchos a muchos entre hechos y tablas de dimensiones. Además de las soluciones descritas en la Sección 3.2.2, estos autores han propuesto otras soluciones que cuentan instancias en lugar de agregar las medidas de una tabla de hechos.

Algunos productos comerciales, tales como Microsoft Analysis Services 2005 y Oracle OLAP 10g, pueden manejar algunas de las jerarquías presentadas en este capítulo, según lo describimos en la Sección 3.7. Por otra parte, IBM ofrece algunas otras herramientas para implementar almacenes de datos y aplicaciones analíticas. Alphablox Analytics [112] se utiliza para el análisis multidimensional de datos representados en un servidor DB2 relacional como esquema de estrella, esquema de copo de nieve y una combinación de ambos. Alphablox Analytics permite definir jerarquías balanceadas, no balanceadas y no cubiertas, donde las últimas dos pueden implementarse como una tabla plana con valores nulos para los miembros ausentes. Adicionalmente, las jerarquías no balanceadas que representan relaciones padre-hijo recursivas pueden almacenarse en una tabla padre-hijo. Alphablox Analytics no permite definir jerarquías alternativas ni paralelas.

3.9 Resumen

Los almacenes de datos se definen utilizando una representación multidimensional de datos, basada en los conceptos de hechos, medidas, dimensiones y jerarquías. Los sistemas OLAP permiten a los usuarios realizar consultas de forma interactiva en los almacenes de datos utilizando operaciones como la navegación ascendente y la descendente, y estas operaciones requieren la definición de jerarquías para agregar medidas.

Una jerarquía representa algún tipo de estructura organizacional, geográfica o de otra índole que sea relevante para propósitos de análisis. No obstante, si bien en las aplicaciones de la vida real se pueden encontrar muchos tipos de jerarquías, los sistemas OLAP actuales pueden manejar sólo un número limitado de ellas. Esto impone importantes restricciones cuando los usuarios requieren llevar a cabo análisis multidimensionales complejos, incluidos diversos tipos de jerarquías. De esta manera, los diseñadores deben emplear varias estrategias de implementación para transformar algunos tipos de jerarquías en unas más simples.

Este capítulo ha presentado el modelo MultiDim, un modelo multidimensional conceptual. El modelo MultiDim está basado en el modelo entidad-relación y brinda una notación gráfica intuitiva. Es bien sabido que las representaciones gráficas ayudan a usuarios y diseñadores a entender mejor los requisitos de aplicación. El modelo también tiene una notación textual asociada, descrita en el Apéndice A, el cual además brinda una formalización del modelo.

Hemos clasificado las jerarquías tomando en cuenta sus diferencias a nivel de esquema e instancias y hemos distinguido entre jerarquías simples y alternativas. Estas últimas están compuestas por una o más jerarquías simples que utilizan el mismo criterio de análisis. Asimismo, las jerarquías simples pueden ser de varios tipos: balanceadas, no balanceadas y generalizadas. Las jerarquías recursivas y no cubiertas son casos especiales de jerarquías no balanceadas y generalizadas, respectivamente. Para cada una de estas jerarquías simples se puede aplicar otra especialización para convertirlas en estrictas o no estrictas. Estas últimas admiten relaciones muchos a muchos entre los niveles padre e hijo, definiendo gráficos a nivel de instancias. Por último, hemos examinado la situación en la cual varias jerarquías que emplean diferentes criterios de análisis podrían estar ligadas a la misma dimensión. Dependiendo de si comparten o no niveles, las hemos llamado jerarquías paralelas dependientes y jerarquías paralelas independientes, respectivamente.

Asimismo, hemos brindado un mapeo de nuestro modelo conceptual a los modelos relacionales y objeto-relacionales, para lo cual utilizamos reglas muy conocidas que transforman los modelos entidad-relación a modelos lógicos. Hemos propuesto algunas modificaciones a estas reglas para

el caso de las jerarquías generalizadas y no estrictas. También hemos comparado enfoques relacionales alternativos para implementar algunos tipos de jerarquías y hemos dado indicaciones con respecto a cuándo utilizar estos enfoques alternativos. Por último, hemos explicado cómo implementar los diferentes tipos de jerarquías en productos comerciales, utilizando como ejemplos Microsoft Analysis 2005 y Oracle OLAP 10g.

La notación del modelo MultiDim permite hacer una clara distinción entre cada tipo de jerarquía, lo cual no sucede con la mayoría de los modelos multidimensionales conceptuales existentes. Es importante representar esta diferenciación a nivel conceptual, puesto que al realizar un mapeo al nivel lógico no es posible distinguir entre jerarquías generalizadas, alternativas y paralelas dependientes. Por lo tanto, el modelo MultiDim nos permite captar de mejor manera la semántica multidimensional de los datos, lo cual puede aprovecharse luego en los sistemas de almacenes de datos y OLAP.

Preguntas de repaso

3.1 Analice los siguientes conceptos: dimensión, nivel, atributo, llave, relación de hechos, rol, medida, jerarquía, relación padre-hijo, cardinalidades, nivel raíz, nivel hoja y criterio de análisis.

3.2 Explique la diferencia, a nivel de esquema y de instancias, entre las jerarquías balanceadas y no balanceadas.

3.3 Brinde un ejemplo de jerarquía recursiva. Explique cómo representar una jerarquía no balanceada con una recursiva.

3.4 Explique la utilidad de las jerarquías generalizadas. ¿A cuál concepto del modelo entidad-relación están relacionadas estas jerarquías?

3.5 ¿Qué es un nivel de separación? ¿Qué es un nivel de unión? ¿Una jerarquía generalizada tiene siempre un nivel de unión?

3.6 Explique por qué las jerarquías no cubiertas (o desiguales–*ragged*) son un caso particular de las jerarquías generalizadas.

3.7 Utilice un ejemplo para describir cómo el tipo de una generalización en el modelo ER (total vs. parcial, exclusiva vs. solapada) influye sobre la jerarquía generalizada correspondiente en el modelo multidimensional.

3.8 ¿Cuál es la diferencia entre jerarquías estrictas y no estrictas?

3.9 Ilustre con un ejemplo el problema de conteo doble de las medidas para las jerarquías no estrictas. Describa diferentes soluciones a este problema.

3.10 ¿Qué es un factor de distribución? Explique la importancia de escoger un factor de distribución apropiado.

3.11 Explique en qué situaciones se emplean las jerarquías alternativas?

3.12 Describa la diferencia entre jerarquías paralelas dependientes y paralelas independientes.

3.13 Ilustre con ejemplos la diferencia entre jerarquías generalizadas, alternativas y paralelas.

3.14 ¿Cuáles son las dimensiones de juegan diferentes roles?

3.15 ¿Por qué a algunas dimensiones se les conoce como dimensiones de hechos o degeneradas?

3.16 Relacione el problema del conteo doble con las dependencias funcionales y multivaluadas contenidas en una relación de hechos.

3.17 ¿Por qué una relación de hechos debe descomponerse cuando manejamos dependencias? Dé un ejemplo de una relación de hechos que pueda descomponerse de otra forma según las dependencias que la contienen.

3.18 Analice las reglas de mapeo para convertir un esquema MultiDim a uno relacional. ¿Se asemejan estas reglas a aquellas utilizadas para transformar un esquema ER a uno 4

3.19 Esplique cómo una jerarquía balanceada puede mapearse en tablas normalizadas o desnormalizadas. Analice las ventajas y desventajas de estos mapeos alternativos.

3.20 ¿Cómo transformar a nivel lógico una jerarquía no balanceada en una balanceada?

3.21 Describa diferentes enfoques para representar jerarquías generalizadas a nivel lógico.

3.22 Explique cómo una jerarquía no estricta puede representarse en el modelo relacional y en el modelo objeto relacional.

3.23 ¿Es posible distinguir entre jerarquías generalizadas, alternativas o paralelas dependientes a nivel lógico?

3.24 Identifique el tipo de jerarquías que pueden representarse directamente en Analysis Services y en Oracle OLAP.

ANEXO 5 – CHAPTER III "CONVENTIONAL DATA WAREHOUSES"

3

Conventional Data Warehouses

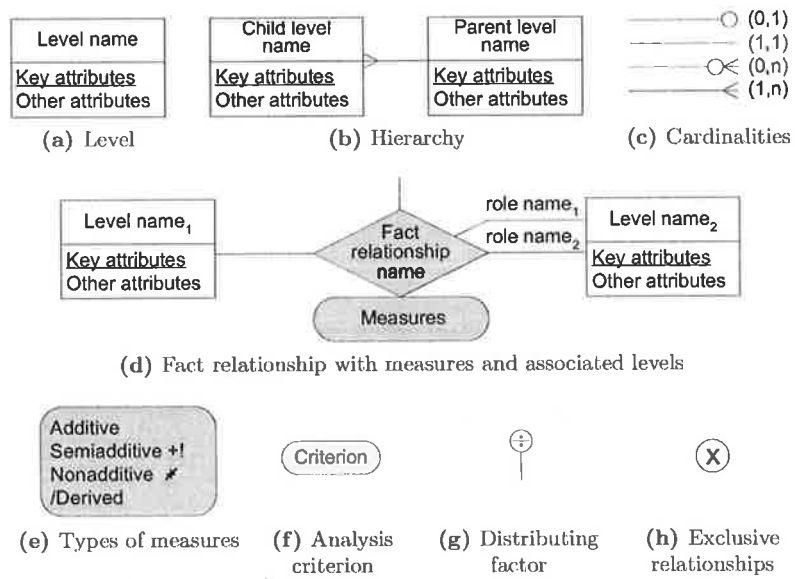
The advantages of using conceptual models for designing applications are well known. In particular, conceptual models facilitate communication between users and designers, since they do not require knowledge about specific features of the underlying implementation platform. Further, schemas developed using conceptual models can be mapped to various logical models, such as relational, object-relational, or object-oriented models, thus simplifying responses to changes in the technology used. Moreover, conceptual models facilitate the maintenance and evolution of applications, since they focus on users' requirements; as a consequence, they provide better support for subsequent changes in the logical and implementation schemas.

In this chapter, we present the MultiDim model, a conceptual multidimensional model that allows one to represent data requirements for data warehouse and OLAP applications. The definition of the model is given in Sect. 3.1. Since hierarchies are essential for exploiting data warehouse and OLAP systems to their full capabilities, in Sect. 3.2 we consider various kinds of hierarchies that exist in real-world situations. We classify these hierarchies, giving a graphical representation of them and emphasizing the differences between some kinds of hierarchies that are currently ignored in the literature. Section 3.3 discusses some advanced aspects of modeling, namely complex hierarchies, and role-playing, fact, and multivalued dimensions. The metamodel of our conceptual multidimensional model is given in Sect. 3.4.

We then give a mapping of our conceptual model to the relational and object-relational models, thus showing the feasibility of implementing our model in current database management systems. We first describe in Sect. 3.5 some general mapping rules for the conceptual model. Then, we discuss in Sect. 3.6 specific mappings for the various kinds of hierarchies, comparing alternative implementations for some of them. Section 3.7 refers to the implementation of hierarchies in commercial platforms, using as examples Microsoft Analysis Services 2005 and Oracle OLAP 10g. Finally, Sect. 3.8 refers to work related to multidimensional conceptual models and hierarchies, and Sect. 3.9 summarizes this chapter.

3.1 MultiDim: A Conceptual Multidimensional Model

In this section, we define the MultiDim model, a conceptual multidimensional model. Our model allows one to represent at the conceptual level all elements required in data warehouse and OLAP applications, i.e., dimensions, hierarchies, and facts with associated measures. The graphical notation of the MultiDim model is shown in Fig. 3.1;[1] it is similar to that of the entity-relationship (ER) model.



(a) Level (b) Hierarchy (c) Cardinalities

(d) Fact relationship with measures and associated levels

(e) Types of measures (f) Analysis criterion (g) Distributing factor (h) Exclusive relationships

Fig. 3.1. Notation of the MultiDim model

In order to give a general overview of the model, we shall use the example in Fig. 3.2, which illustrates the conceptual schema of a sales data warehouse. This figure includes several types of hierarchies, which will be presented in more detail in the subsequent sections.

A **schema** is composed of a set of dimensions and a set of fact relationships.

A **dimension** is an abstract concept that groups data that shares a common semantic meaning within the domain being modeled. A dimension is composed of a set of hierarchies (see later), and a hierarchy is in turn composed of a set of levels.

[1] A more detailed description of our notation is given in Appendix B.

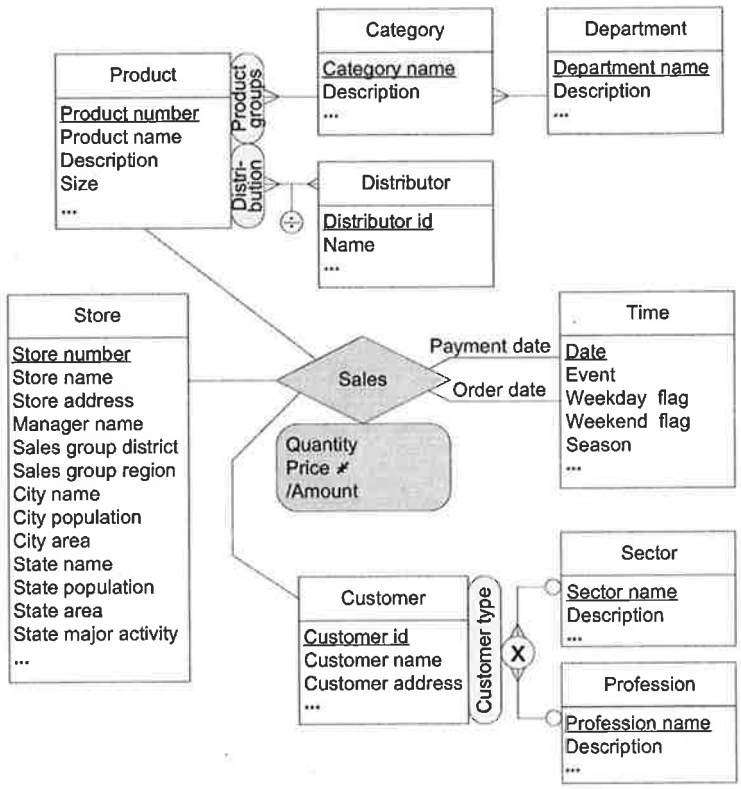


Fig. 3.2. A conceptual multidimensional schema of a sales data warehouse

A **level** corresponds to an entity type in the ER model. It describes a set of real-world concepts that, from the application's perspective, have similar characteristics. For example, Product, Category, and Department are some of the levels in Fig. 3.2. Instances of a level are called **members**. As shown in Fig. 3.1a, a level has a set of **attributes** that describe the characteristics of their members. In addition, a level has one or several **keys** that identify uniquely the members of a level, each key being composed of one or several attributes. For example, in Fig. 3.2, Category name is a key attribute of the Category level. Each attribute of a level has a type, i.e., a domain for its values. Typical value domains are integer, real, and string. For brevity, we do not include type information for attributes in the graphical representation of our conceptual schemas. This can be done if necessary, and it is done in the textual representation of our model, which is shown in Appendix A.

A **fact relationship** (Fig. 3.1d) expresses a focus of analysis and represents an n-ary relationship between levels. For example, the Sales fact relationship between the Product, Store, Customer, and Time levels in Fig. 3.2 is used for analyzing sales figures. Instances of a fact relationship are called **facts**. Since the cardinality of every level participating in a fact relationship is (0,n), we omit such cardinalities to simplify the model. Further, as shown in Fig. 3.1d, the same level can participate several times in a fact relationship, playing different **roles**. Each role is identified by a name and is represented by a separate link between the corresponding level and the fact relationship. For example, in Fig. 3.2 the Time level participates in the Sales fact relationship with the roles Payment date and Order date.

A fact relationship may contain attributes commonly called **measures**. These contain data (usually numerical) that is analyzed using the various perspectives represented by the dimensions. For example, the Sales fact relationship in Fig. 3.2 includes the measures Quantity, Price, and Amount. The key attributes of the levels involved in a fact relationship indicate the granularity of the measures, i.e., the level of detail at which measures are represented.

In Sect. 2.6.2 we classified measures as **additive**, **semiadditive**, or **nonadditive**. As shown in Fig. 3.1e, we assume by default that measures are additive, i.e., they can be summarized along all dimensions. For semiadditive and nonadditive measures, we include the symbols +! and $\neq$, respectively, next to the measure's name. Further, measures and level attributes may be **derived**, when they are calculated on the basis of other measures or attributes in the schema. We use the symbol / for indicating derived attributes and measures. For example, Fig. 3.2 the measures Quantity, Price, and Amount are, respectively, additive, semi-additive, and derived measures.

Dimensions are used for analyzing measures from various perspectives. A dimension is composed of either one level or one or more **hierarchies**, which establish meaningful aggregation paths. A hierarchy comprises several related levels, as in Fig. 3.1b. Given two related levels of a hierarchy, the lower level is called the **child** and the higher level is called the **parent**. Thus, the relationships composing hierarchies are called **parent-child relationships**. Since these relationships are used only for traversing from one level to the next, they are simply represented by a line to simplify the notation.

Parent-child relationships are characterized by **cardinalities**, shown in Fig. 3.1c, indicating the minimum and the maximum number of members in one level that can be related to a member in another level. For example, in Fig. 3.2 the child level Product is related to the parent level Category with a one-to-many cardinality, which means that every product belongs to only one category and that each category can have many products.

The hierarchies in a dimension may express various structures used for analysis purposes; thus, we include an **analysis criterion** (Fig. 3.1f) to differentiate them. For example, the Product dimension in Fig. 3.2 includes two hierarchies: Product groups and Distribution. The former hierarchy comprises the levels Product, Category, and Department, while the latter hierarchy in-

cludes the levels Product and Distributor. If a single level contains attributes forming a hierarchy, such as City name and State name in the Store dimension in Fig. 3.2, this means that the user is not interested in employing this hierarchy for aggregation purposes.

The levels in a hierarchy are used to analyze data at various **granularities**, i.e., levels of detail. For example, the Product level contains specific information about products, while the Category level may be used to see these products from the more general perspective of the categories to which they belong. The level in a hierarchy that contains the most detailed data is called the **leaf level**; it must be the same for all hierarchies included in a dimension. The leaf-level name is used for defining the dimension's name. The last level in a hierarchy, representing the most general data, is called the **root level**. If several hierarchies are included in a dimension, their root levels may be different. For example, both hierarchies in the Product dimension in Fig. 3.2 contain the same leaf level, Product, while they have different root levels, the Department and the Distributor levels.

In some publications, the root of a hierarchy is represented using a level called All. We leave to designers the decision of including this level in multidimensional schemas. In this book, we do not present the All level for the various hierarchies, since we consider that it is meaningless in conceptual schemas and, in addition, it adds unnecessary complexity to them.

The key attributes of a parent level define how child members are grouped. For example, in Fig. 3.2, Department name in the Department level is a key attribute; it is used for grouping different category members during the roll-up operation from the Category to the Department level. However, in the case of many-to-many parent-child relationships, it is also needed to determine how to distribute the measures from a child to its parent members. For example, in Fig. 3.2 the relationship between Product and Distributor is many-to-many, i.e., the same product can be distributed by several distributors. The notation in Fig. 3.1g indicates that a distributing factor is used to divide the measures associated with a product between its categories.

Finally, it is sometimes the case that two or more parent-child relationships are **exclusive**. This is represented using the symbol in Fig. 3.1h. An example is given in Fig. 3.2, where customers can be either persons or organizations. Thus, according to their type, customers participate in only one of the relationships departing from the Customer level: persons are related to the Profession level, while organizations are related to the Sector level.

3.2 Data Warehouse Hierarchies

Hierarchies are fundamental in analytical applications, since they provide the means to represent the parameters under analysis at different abstraction levels. In real-world situations, users must deal with complex hierarchies of various kinds. However, the logical models of current data warehouse and

OLAP systems allow only a limited set of kinds of hierarchies. Therefore, users are often unable to capture the essential semantics of multidimensional applications and must limit their analysis to considering only the predefined set of hierarchies provided by the tools in use. Further, current approaches for representing hierarchies at the conceptual level focus mainly on aggregation paths represented at the schema level, i.e., they focus on establishing sequences of levels that should be traversed during roll-up and drill-down operations. However, as we shall see in this section, the distinction between the various kinds of hierarchies should also be made at the instance level, i.e., we should consider the cardinalities in parent-child relationships.

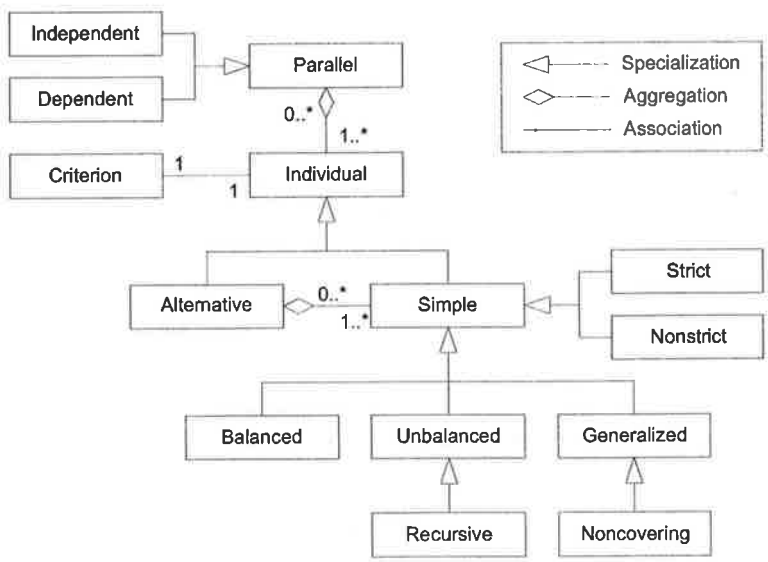


Fig. 3.3. Hierarchy classification

In this section, we discuss various kinds of hierarchies that can be represented in the MultiDim model. To give a more general view of our proposal for the various kinds of hierarchies and the relationships between them, we first present a classification of them using the metamodel shown in Fig. 3.3. As can be seen in the figure, parallel hierarchies can be specialized into independent and dependent hierarchies, depending on whether they share common levels. A parallel hierarchy is an aggregation of individual hierarchies. Each individual hierarchy is associated with one analysis criterion. An individual hierarchy can be simple or alternative, where the latter is composed of one or several simple hierarchies associated with the same analysis criterion. The simple hierarchies include further kinds: balanced, unbalanced, and general-

ized hierarchies. Also, recursive and noncovering hierarchies are a special case of unbalanced and generalized hierarchies, respectively. For each of these simple hierarchies, another specialization can be applied, depending on whether the cardinalities between the parent and child levels are one-to-many or many-to-many. The former are called strict hierarchies, while the latter are called nonstrict hierarchies.

In the following, we refer in more detail to the various kinds of hierarchies, presenting them at the conceptual and at the logical level. Their implementation at the physical level, using as examples Microsoft Analysis Services 2005 and Oracle OLAP 10g, is covered in Sect. 3.7.

3.2.1 Simple Hierarchies

Simple hierarchies are those hierarchies where, if all its component parent-child relationships are one-to-many, the relationship between their members can be represented as a tree. Further, these hierarchies use only one criterion for analysis. Simple hierarchies are divided into balanced, unbalanced, and generalized hierarchies.

Balanced Hierarchies

A **balanced hierarchy** has only one path at the schema level (Fig. 3.4a). At the instance level, the members form a tree where all the branches have the same length (Fig. 3.4b). As implied by the cardinalities, all parent members have at least one child member, and a child member belongs to only one parent member. For example, in Fig. 3.4 each category has assigned to it at least one product, and a product belongs to only one category.

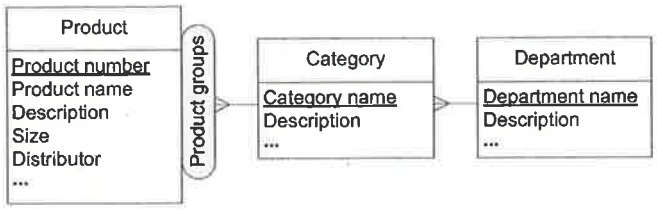
Unbalanced Hierarchies

An **unbalanced hierarchy**[2] has only one path at the schema level. However, as implied by the cardinalities, at the instance level some parent members may not have associated child members. Figure 3.5a shows a hierarchy in which a bank is composed of several branches: some of them have agencies with ATMs, some have only agencies, and small branches do not have any organizational division. As a consequence, at the instance level the members represent an unbalanced tree (Fig. 3.5b), i.e., the branches of the tree have different lengths, since some parent members do not have associated child members. As in the case of balanced hierarchies, the cardinalities imply that every child member should belong to at most one parent member. For example, in Fig. 3.5 every agency belongs to one branch.

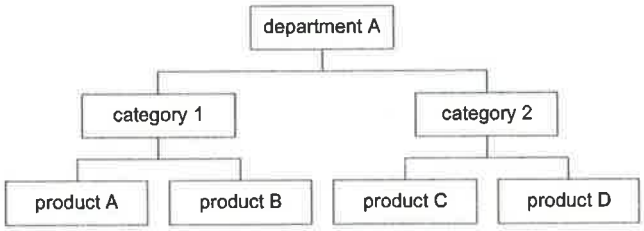
Unbalanced hierarchies include a special case that we call **recursive hierarchies**.[3] In this kind of hierarchy the same level is linked by the two roles of

[2] These hierarchies are also called heterogeneous [107] and non-onto [233] hierarchies.

[3] These are also called parent-child hierarchies [157, 208].



(a) Schema



(b) Example of instances

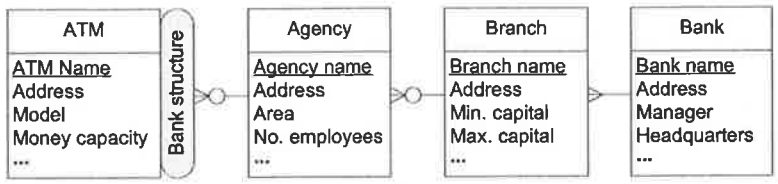
Fig. 3.4. A balanced hierarchy

a parent-child relationship. An example is given in Fig. 3.6a, which represents an organizational chart in terms of the employee-supervisor relationship. The subordinate and supervisor roles of the parent-child relationship are linked to the Employee level. Recursive hierarchies are mostly used when all hierarchy levels express the same semantics, i.e., when the characteristics of the children and parents are similar (or the same), as in Fig. 3.6a, where an employee has a supervisor who is also an employee.

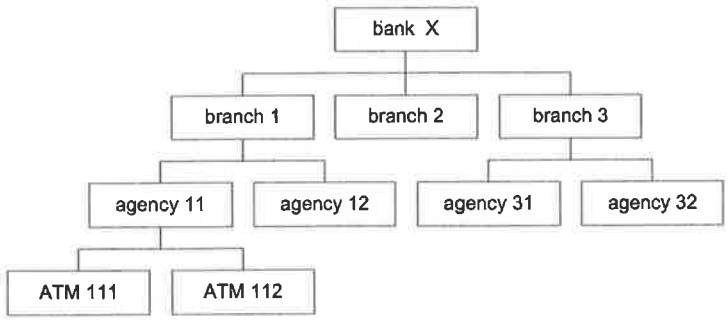
As shown in Fig. 3.6b, a recursive hierarchy could be used to represent the unbalanced hierarchy shown in Fig. 3.5a. However, in this representation the semantics of the hierarchy is somewhat lost, since levels corresponding to concepts at different granularities are represented together, for instance ATM, agency, and branch. Note that an additional discriminating attribute, Entity type, is needed to identify the level to which the members belong. Further, the structure of the hierarchy is not clear and it is necessary to retrieve all members to reconstruct it, i.e., we must move to the instance level to recover information about the schema. In addition, the representation of this hierarchy shown in Fig. 3.6b is less expressive than that in Fig. 3.5a, since the grouping of attributes according to the levels to which they belong is lost.

Generalized Hierarchies

Sometimes a dimension includes subtypes that can be represented by a generalization/specialization relationship [5, 8, 161, 170]. Moreover, the specialized

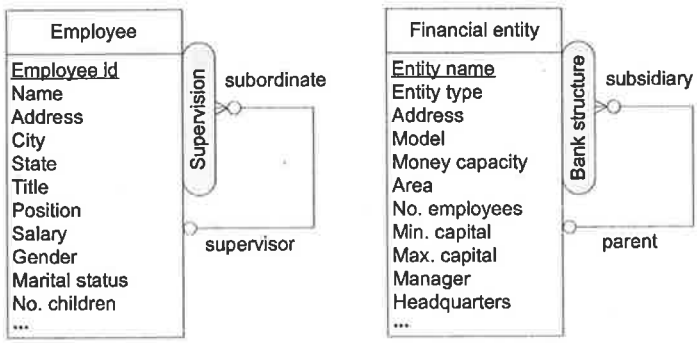


(a) Schema



(b) Examples of instances

Fig. 3.5. An unbalanced hierarchy



(a) (b)

Fig. 3.6. Examples of recursive hierarchies

subtypes can include their own hierarchy. In the example given in Fig. 3.7, the Customer supertype is specialized into the Person and the Company subtypes to distinguish the two types of customers. Further, the subtypes have hierarchies with common levels (for example Branch and Area) and specific levels (for example Profession and Class for Person, and Type and Sector for Company).

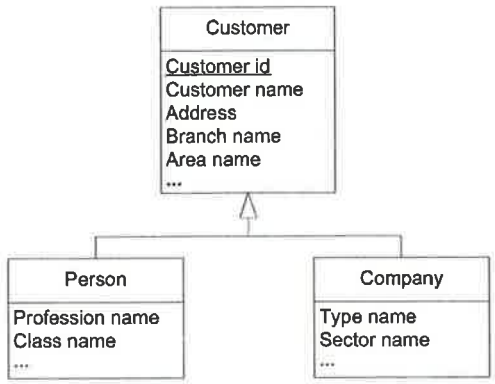
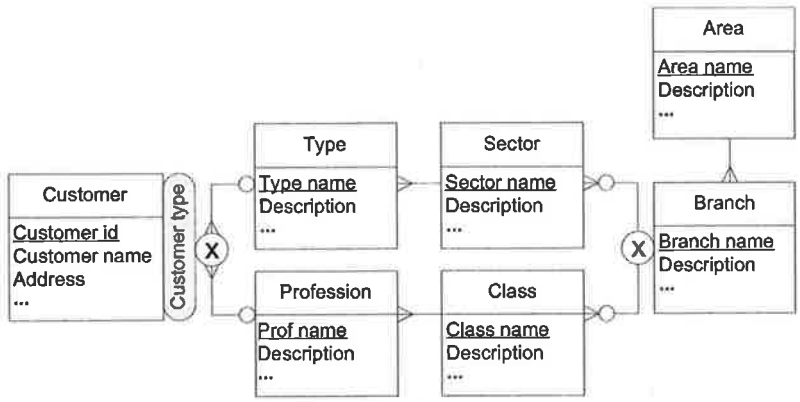


Fig. 3.7. Entity-relationship representation of customer types

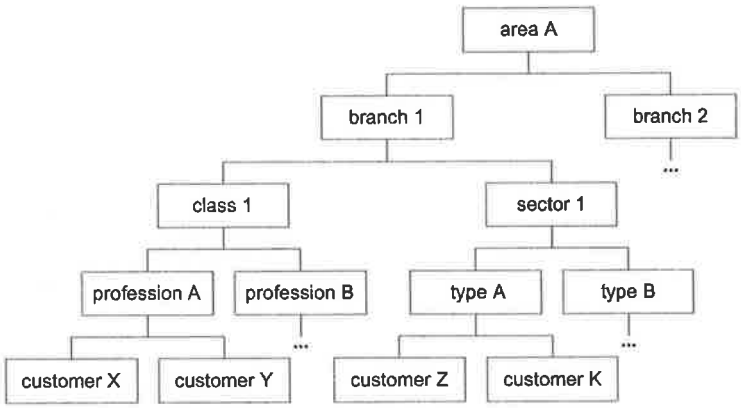
In the ER representation, the hierarchical paths cannot be clearly distinguished, in particular because higher hierarchy levels (for example Area) in a supertype are not related to the other hierarchy levels in its subtypes. Therefore, the information about the levels forming a hierarchy and the parent-child relationships between them cannot be retrieved. For example, in Fig. 3.7 it is not clear that measures related to a customer that is a person can be aggregated using a hierarchy formed by the levels Profession–Class–Branch–Area. Further, while both hierarchies can be represented independently by repeating the common levels, the complexity of the schema is reduced if it is possible to include shared levels characterized by the same granularity, such as the Branch and Area levels in our example. To represent such kinds of hierarchies, we propose the graphical notation shown in Fig. 3.8a, where the common and specific hierarchy levels and also the parent-child relationships between them are clearly represented. We call such hierarchies **generalized hierarchies**.

At the schema level, a generalized hierarchy contains multiple exclusive paths sharing at least the leaf level; they may also share some other levels, as shown in Fig. 3.8a. All these paths represent one hierarchy and account for the same analysis criterion. At the instance level, each member of the hierarchy belongs to only one path, as can be seen in Fig. 3.8b.[4] We use the symbol $\otimes$ to indicate that the paths are exclusive for every member. Such a notation is

[4] Note that, as indicated by the ellipses, not all members are represented in the figure.



(a) Schema



(b) Examples of instances

Fig. 3.8. A generalized hierarchy

equivalent to the **xor** annotation used in UML [30].[5] The levels at which the alternative paths split and join are called the **splitting and joining levels**, respectively.

In the example in Fig. 3.8a, the paths between the two ⊗ symbols refer to attributes that can be used for aggregation purposes for the specialized subtypes shown in Fig. 3.7. The lower path (i.e., the path containing the Profession and Class levels) corresponds to the Person subtype, while the upper path (i.e., that containing the Type and Sector levels) indicates the Company subtype.

[5] This notation is also used by Hurtado et al. [106] to denote the "splitting constraints".

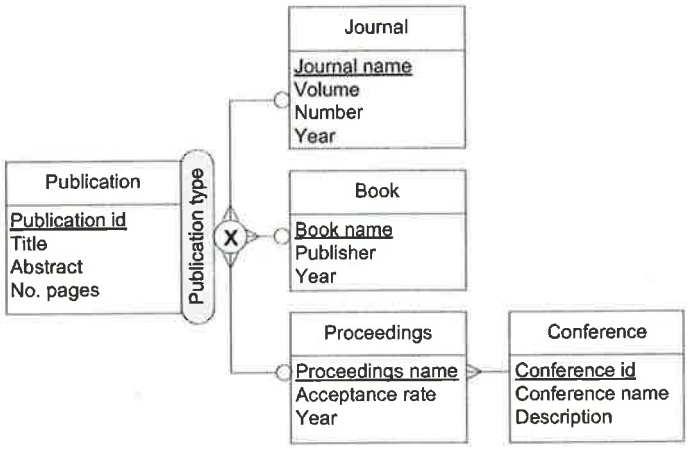


Fig. 3.9. A generalized hierarchy without a joining level

Note that the supertype included in the generalization/specialization relationship in Fig. 3.7 is used in generalized hierarchies for representing a leaf level. However, it will include only those attributes that represent concepts at the lowest granularity (for example Customer id, Customer name, and Address). This is required to ensure that all hierarchy members can be linked to the same fact relationship and to avoid the possibility that each subtype will be handled as a separate dimension (for instance, separate dimensions for the Person and the Company subtypes) with different fact relationships.

Further, the distinction between splitting and joining levels in generalized hierarchies is important for correct measure aggregation during roll-up and drill-down operations [106]. This kind of hierarchy does not satisfy the summarizability conditions [164] (see Sect. 2.6.2), because the mapping from the splitting level to the parent levels is incomplete; for example, not all customers roll up to the Profession level. Thus, the aggregation mechanism should be modified when a splitting level is reached in a roll-up operation. Moreover, summarizability conditions can also vary for a joining level. To establish correct aggregation procedures, the approach proposed by Hurtado et al. [106] for heterogeneous hierarchies could be applied. On the other hand, the traditional approach can be used for aggregating measures for common hierarchy levels.

In generalized hierarchies, it is not necessary that splitting levels must be joined. For example, Fig. 3.9 shows a generalized hierarchy used for analyzing international publications. Three kinds of publications are considered, i.e.,

journals, books, and conference proceedings. The latter can be aggregated to the conference level. However, there is not a common joining level for all paths.

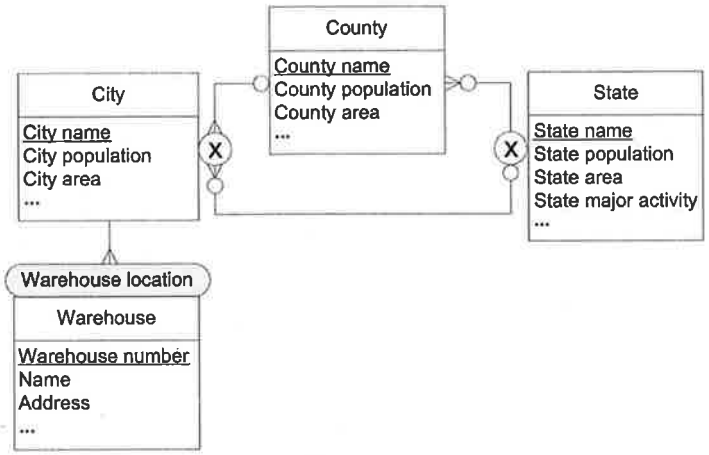
Generalized hierarchies include a special case commonly referred to as **noncovering hierarchies** or **ragged hierarchies** [157, 233]. The example given in Fig. 3.10 represents a distribution company that has warehouses in several different states. However, the geographical division in these states may vary; for example, the division into counties may be skipped for some of them. A noncovering hierarchy is a generalized hierarchy with the additional restriction that the alternative paths are obtained by skipping one or several intermediate levels. At the instance level, every child member has only one parent member, although the path length from the leaves to the same parent level can be different for different members.

Note that not all generalization/specialization hierarchies can be represented by generalized hierarchies in a multidimensional model. Recall that a generalization/specialization can be (a) total or partial, and (b) disjoint or overlapping [67] (see Sect. 2.2). Partial specializations induce an additional path in the generalized hierarchy that relates the common levels. For example, if the generalization in Fig. 3.7 were partial, indicating that there were some customers that were considered to be neither persons nor companies, then an additional path between the Customer and Branch levels would be needed in the hierarchy of Fig. 3.8a.

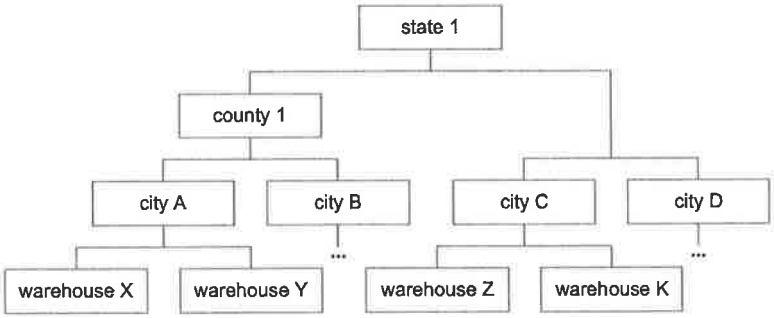
On the other hand, for overlapping generalizations, various options are possible, according to the users' requirements and the availability of measures. Consider, for example, an overlapping generalization where a person who owns a company buys products either for his/her individual use or for the company. If measures are known only for the superclass Customer, then only the hierarchy with common levels will be represented, for instance the Customer, Branch, and Area levels in Fig. 3.8a. If measures are known only for each subclass, for instance for Person and Company, separate dimensions and fact relationships with corresponding measures can be created for each specialization. However, significant programming effort is required to manage dimensions with overlapping sets of members and to develop aggregation procedures that allow common hierarchy levels to be traversed. Another solution is to disallow overlapping generalizations for multidimensional hierarchies.

3.2.2 Nonstrict Hierarchies

For the simple hierarchies presented above, we assumed that each parent-child relationship has a one-to-many cardinality, i.e., a child member is related to at most one parent member and a parent member may be related to several child members. However, many-to-many relationships between parent and child levels are very common in real-life applications. For example, a diagnosis may



(a) Schema



(b) Examples of instances

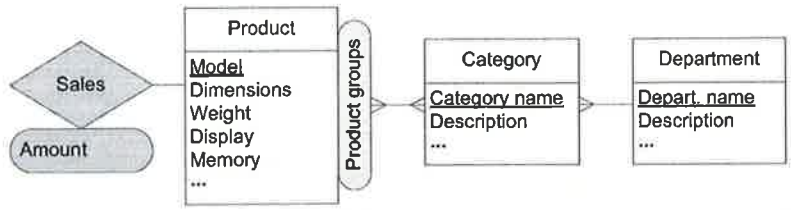
Fig. 3.10. A noncovering hierarchy

belong to several diagnosis groups [233], a week may span two months, a mobile phone can be classified into various product categories, etc.

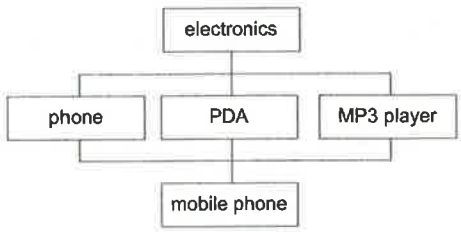
We call a hierarchy **nonstrict** if, at the schema level, it has at least one many-to-many relationship; it is called **strict** if all relationships have one-to-many cardinalities. The fact that a hierarchy is strict or not is orthogonal to its kind. Thus, the various kinds of hierarchies previously presented can be either strict or nonstrict.

Figure 3.11a shows a balanced nonstrict hierarchy where a product may belong to several different categories.[6] For example, a mobile phone may not only provide calling facilities but also may serve as a personal digital assis-

[6] This example is inspired by [104].



(a) Schema



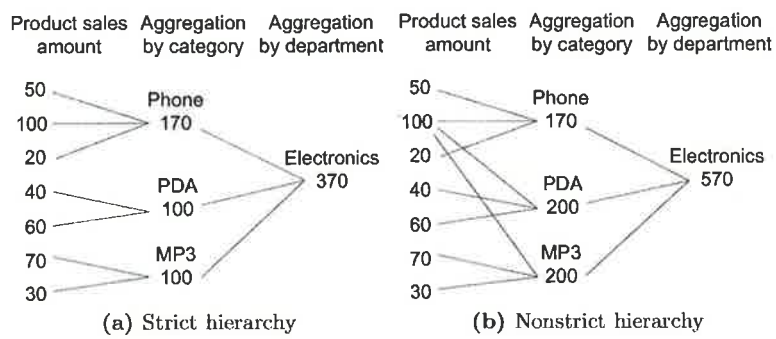
(b) Examples of instances

Fig. 3.11. A balanced nonstrict hierarchy

tant (PDA) and as an MP3 player. In a strict hierarchy, this product can only belong to one category, i.e., "phone". In a nonstrict hierarchy it may be considered as belonging to three categories, i.e., "phone", "PDA", and "MP3 player". Therefore, since at the instance level a child member may have more than one parent member, the members of the hierarchy form an acyclic graph (Fig. 3.11b).

Note the slight abuse of terminology. We use the term "nonstrict hierarchy" to denote an acyclic classification graph [104]. We use this term for several reasons. Firstly, the term "hierarchy" conveys the notion that users need to analyze measures at different levels of detail, which is less clear with the term "acyclic classification graph". Secondly, the term "hierarchy" is already used by practitioners and, as we shall see in the next section, some tools, in particular SQL Server Analysis Services 2005, allow many-to-many parent-child relationships. Finally, the term " hierarchy" is also used by several researchers (e.g., [5, 170, 233, 302]).

Nonstrict hierarchies induce the problem of **double counting** of measures when a roll-up operation reaches a many-to-many relationship. Let us consider the example in Fig. 3.12. This illustrates sales of products with aggregations according to the Category and Department levels. Suppose that the product with sales equal to 100 is a mobile phone that also includes PDA and MP3 player functionalities. In a strict hierarchy, this mobile phone can belong to only one category. Therefore, as shown in Fig. 3.12a, the sum of sales by category and by department can be calculated straightforwardly. However,



(a) Strict hierarchy (b) Nonstrict hierarchy

Fig. 3.12. Example of aggregation for a sales amount measure

it might be interesting also to consider this mobile phone when calculating the sum of sales for PDAs and MP3 players. Nevertheless, as can be seen in Fig. 3.12b, this approach causes incorrect aggregated results, since the phone's sales are counted three times instead of only once. Note that incorrect results will propagate when higher hierarchy levels are traversed, for example Department in Fig. 3.11a.

One solution to the problem of double counting of measures, proposed by Pedersen et al. [233], is to transform a nonstrict hierarchy into a strict hierarchy by creating a new member that joins into one group the parent members participating in a many-to-many relationship. In our mobile phone example, a new category member will be created that represents the three categories together: mobile phone, PDA, and MP3 player. Another solution is to ignore the existence of several parent members and to choose one of them as the primary member [203]; for example, we may choose only the mobile phone category. However, neither of these solutions may correspond to the users' analysis requirements, since in the former, artificial categories are introduced, and in the latter, some pertinent analysis scenarios are ignored.

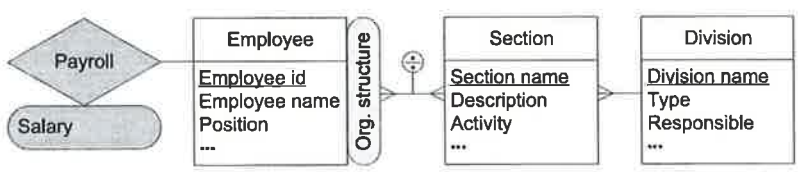


Fig. 3.13. A nonstrict hierarchy with a distributing factor

An alternative approach to the double-counting problem can be taken by indicating how measures are distributed between several parent members for

many-to-many relationships. For example, Fig. 3.13 shows a nonstrict hierarchy where employees may work in several sections.[7] The schema includes a measure that represents an employee's overall salary, i.e., the sum of the salaries paid in each section. It may be the case that the percentage of time for which an employee works in each section is known. Therefore, in this case we include an additional symbol ⊜, indicating that a **distributing factor** determines how measures should be divided between several parent members in a many-to-many relationship. The choice of an appropriate distributing factor is important in order to avoid approximate results. For example, suppose that the distributing factor represents the percentage of time for which an employee works in a specific section. If an employee has a higher position in one section, then even though he/she works less time in that section, he/she may earn a higher salary. Therefore, applying the percentage of time as the distributing factor for measures representing an employee's overall salary may not give an exact result.

Note that in the mobile phone example, distribution of the sales measure between three categories according to previously specified percentages, for example 70% to "phone", 20% to "PDA", and 10% to "MP3", may not give meaningful information to users. In other situations, this distribution is impossible to specify. In the latter case, if a user does not require exact values of measures, this distributing factor can be calculated by considering the total number of parent members with which the child member is associated [283]. In our mobile phone example, since we have three categories with which this phone is associated, 1/3 of the value of the measure will be aggregated for each category.

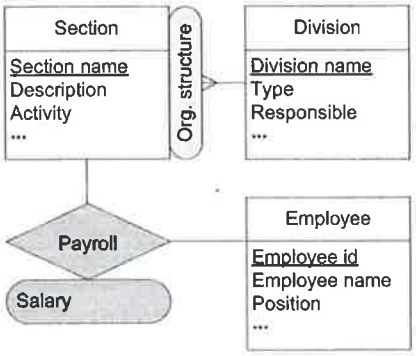


Fig. 3.14. Transforming a nonstrict hierarchy into a strict hierarchy with an additional dimension

[7] For simplicity, we omit in the figure other dimensions, such as the time dimension.

Another solution could be to transform a nonstrict hierarchy into independent dimensions as shown in Fig. 3.14. This solution corresponds to a different conceptual schema, where the focus of analysis has been changed from employee's salaries to employee's salaries by section. Note that this solution can only be applied when the exact distribution of the measures is known, for instance, when the amounts of salary paid for working in the different sections are known. It cannot be applied for nonstrict hierarchies without a distributing factor, as in Fig. 3.11a.

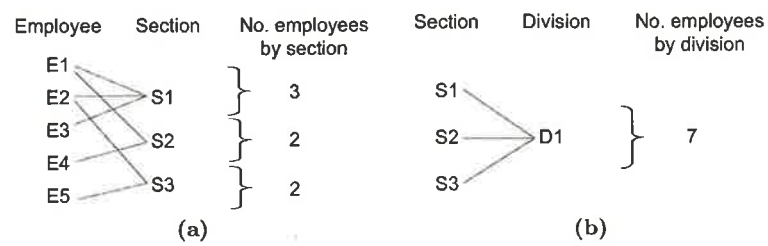


Fig. 3.15. Double-counting problem for a nonstrict hierarchy

Nevertheless, although the solution in Fig. 3.14 allows one to aggregate correctly the salary measure when applying the roll-up operation from the Section to the Division level, the problem of double counting of the same employee will occur. Suppose that we want to use the schema in Fig. 3.14 to calculate the number of employees by section or by division; this value can be calculated by counting the instances of employees in the fact relationship. The example in Fig. 3.15a considers five employees, who are assigned to various sections. Counting the number of employees who work in each section gives correct results. However, the aggregated values for each section cannot be reused for calculating the number of employees in every division, since some employees (E1 and E2 in Fig. 3.15a) will be counted twice and the total result will give a value equal to 7 (Fig. 3.15b) instead of 5.

In summary, nonstrict hierarchies can be handled in several different ways:

- Transforming a nonstrict hierarchy into a strict one:
  - Creating a new parent member for each group of parent members linked to a single child member in a many-to-many relationship [233].
  - Choosing one parent member as the primary member and ignoring the existence of other parent members [203].
  - Splitting the hierarchy in two at the many-to-many relationship, where the levels from the parent level and beyond become a new dimension.
- Including a distributing factor [147].
- Calculating approximate values of a distributing factor [283].

Since each solution has its advantages and disadvantages and requires special aggregation procedures, the designer must select the appropriate solution according to the situation at hand and the user's requirements.

3.2.3 Alternative Hierarchies

Alternative hierarchies represent the situation where, at the schema level, there are several nonexclusive simple hierarchies that share at least the leaf level and account for the same analysis criterion. An example is given in Fig. 3.16a, where the Time dimension includes two hierarchies, corresponding to different subdivisions of years. As seen in Fig. 3.16b,[8] at the instance level such hierarchies form a graph, since a child member can be associated with more than one parent member, and these parent members may belong to different levels.

Alternative hierarchies are needed when the user needs to analyze measures from a unique perspective (for example time) using alternative aggregations. Since the measures in the fact relationship will participate totally in each component hierarchy, measure aggregation can be performed as for simple hierarchies. Aggregated measures for a common hierarchy level can be reused when the various component hierarchies are traversed.

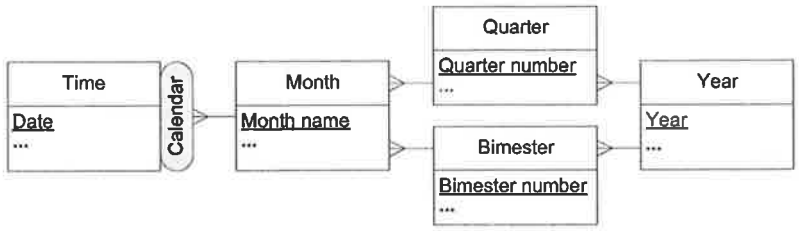
However, in the case of alternative hierarchies it is not semantically correct to simultaneously combine different component hierarchies, since this would result in meaningless intersections [296], i.e., a combination of members that do not have values for aggregated measures. In the present example, this would be the case when combining the members B1-2001 and Q2-2001 or Q2-2001 and Jan-2001. This situation induces problems in the interpretation of query results, since the lack of an aggregated value for measures, for example sales, can be perceived by a user as if there were not sales during the corresponding period, instead of realizing that the combination of levels is invalid. Therefore, users must choose only one of the alternative aggregation paths for their analysis, and switch to the other one if required.

Note the difference between generalized and alternative hierarchies (Figs. 3.8 and 3.16). Although the two kinds of hierarchies share some levels and use only one analysis criterion, they represent different situations. In a generalized hierarchy a child member is related to only one of the paths, whereas in an alternative hierarchy a child member is related to all paths, and the user must choose one of them for analysis.

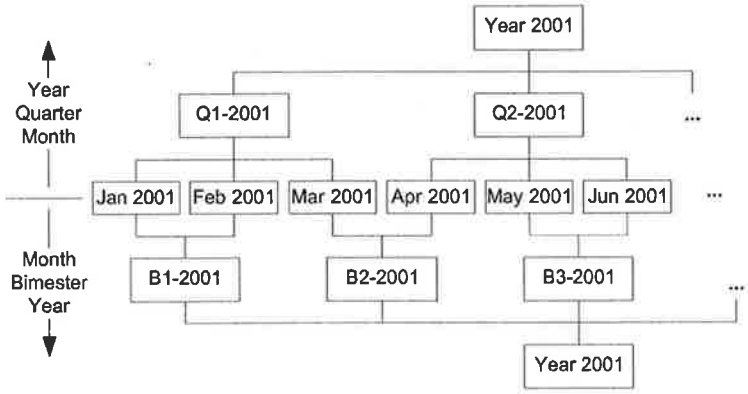
3.2.4 Parallel Hierarchies

Parallel hierarchies arise when a dimension has several hierarchies associated with it, accounting for different analysis criteria. Such hierarchies can be independent or dependent.

[8] Note that in the figure, the members of the Time level are not shown.



(a) Schema



(b) Examples of instances

Fig. 3.16. Alternative hierarchies

Parallel Independent Hierarchies

Parallel independent hierarchies are composed of disjoint hierarchies, i.e., hierarchies that do not share levels. The component hierarchies may be of different kinds. Figure 3.17 shows an example of a dimension that has both a balanced and a noncovering hierarchy, used for different analysis criteria. The hierarchy Product groups is used for grouping products according to categories or departments, while the hierarchy Distributor location groups them according to distributors' divisions or regions.

Parallel Dependent Hierarchies

Parallel dependent hierarchies are composed of several hierarchies that account for different analysis criteria and share some levels. The example given in Fig. 3.18 represents a company that requires sales analysis for stores located in several countries. The Store dimension contains two balanced hierarchies. The hierarchy Store location represents the geographic division of the store

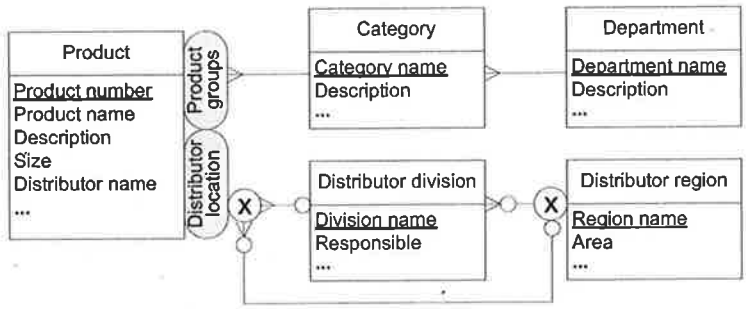


Fig. 3.17. Parallel independent hierarchies, composed of a balanced and a noncovering hierarchy

address and includes the levels Store, City, State, and Country. The hierarchy Sales organization represents the organizational division of the company and includes the levels Store, Sales district, State, and Sales region. Since the two hierarchies share the State level, this level plays different roles according to hierarchy chosen for the analysis. Sharing levels in a conceptual schema allows the designer to reduce the number of its elements, i.e., to improve its readability, without loosing its semantics. Note that in order to unambiguously define the levels composing the various hierarchies, the analysis criteria must be included in the sharing level for hierarchies that continue beyond that level.

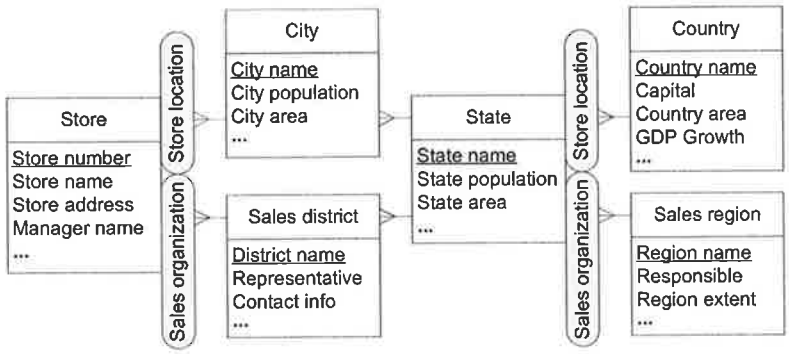


Fig. 3.18. Parallel dependent hierarchies, composed of two balanced hierarchies

Note that even though both alternative and parallel hierarchies share some levels and may include several simple hierarchies, they represent different situations and should be clearly distinguishable at the conceptual level. This is done by including only one (for alternative hierarchies) or several (for parallel

dependent hierarchies) analysis criteria. In this way, the user is aware that in the case of alternative hierarchies it is not meaningful to combine levels from different component hierarchies, whereas this can be done for parallel hierarchies. For example, for the schema in Fig. 3.18 the user can issue a query "what are the sales figures for stores in city A that belong to the sales district X" without producing meaningless intersections.

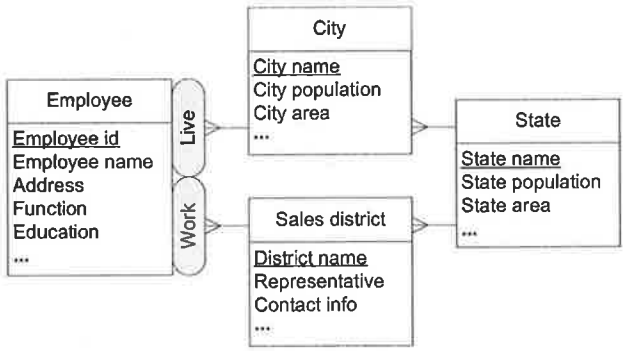


Fig. 3.19. Parallel dependent hierarchies leading to different parent members of the shared level

Further, if the hierarchies composing a set of alternative hierarchies share a level which is not the leaf, then a child member is related to the same parent level in all the alternative hierarchical paths. For example, in the case of Fig. 3.16, traversing the Time hierarchy from a specific day in the Time level will end up in the same year independently of which path is used, i.e., either Time–Month–Quarter–Year or Time–Month–Bimester–Year. The situation is different for parallel dependent hierarchies. For instance, consider the schema in Fig. 3.19, which refers to the living and working places of sales employees. It should be obvious that traversing the hierarchies Live and Work from the Sales employee to the State level will lead to different states for employees who live in one state and work in another.

As a consequence of the difference described above, aggregated measure values can be reused for shared levels in alternative hierarchies, whereas this is not the case for parallel dependent hierarchies. For example, suppose that the amount of sales generated by employees E1, E2, and E3 are €50, €100, and €150, respectively. If all employees live in state A, but only E1 and E2 work in this state, aggregating the sales of all employees to the State level following the Live hierarchy gives a total amount of €300, whereas the corresponding value will be equal to €150 when the Work hierarchy is traversed. Note that both results are correct, since the two hierarchies represent different analysis criteria.

3.3 Advanced Modeling Aspects

In this section, we discuss some particular features of the MultiDim model in order to show its applicability for representing complex multidimensional schemas. We begin by describing some complex hierarchies that can be represented by the model. We then discuss role-playing dimensions, fact dimensions, and multivalued dimensions.

3.3.1 Modeling of Complex Hierarchies

The example in Fig. 3.20 includes all the kinds of hierarchies described in the previous sections. This example shows three fact relationships, indicating that the focus of analysis is related to sales, the payroll, and sales incentive programs. In the figure only the names of the levels are shown; attribute names are not included.

As shown in the figure, the MultiDim model allows us to represent shared levels and shared dimensions. Sharing levels allows existing data to be reused, while sharing dimensions opens the possibility of analyzing measures obtained from different fact relationships using the drill-across operation (see Sect. 2.6.3) [147]. For example, the level Month is shared between two hierarchies (Sales time and Payroll time), while the level State is shared between three hierarchies (Sales organization, Store location, and Customer location). The Store dimension, with two hierarchies Sales organization and Store location, is shared between the Sales and the Payroll fact relationships. Thus, the measure Amount (from the Sales fact relationship) can be analyzed together with Base salary (from the Payroll fact relationship) for different stores and for different periods of time.

Further, the MultiDim model allows one to define several parent-child relationships between the same levels, for example the relationships between Employee and Section for the Works and Affiliated hierarchies. Each relationship corresponds to a different analysis criterion and has an associated set of instances.

Another feature of the MultiDim model is the possibility to represent complex **generalized hierarchies**, which, to our knowledge, are not considered in other models with graphical notations (for example, the model in [110]). The example shown in Fig. 3.21a (inspired by [106]) models the situation where stores belong to several different provinces, but only some provinces are grouped into states, as can be seen at the instance level in Fig. 3.21b. The cardinality (0,1) in Fig. 3.21a indicates that not all members of the Province level will have assigned to them a parent in the State level. This is shown in Fig. 3.21b, where the member all is used to represent the maximum aggregation level that exists for this hierarchy. Figure 3.22 (inspired by [105]) shows another example of a complex generalized hierarchy. This indicates three different aggregation paths that exist for products: Product Brand, Product Brand Category Department, and Product Category Department.

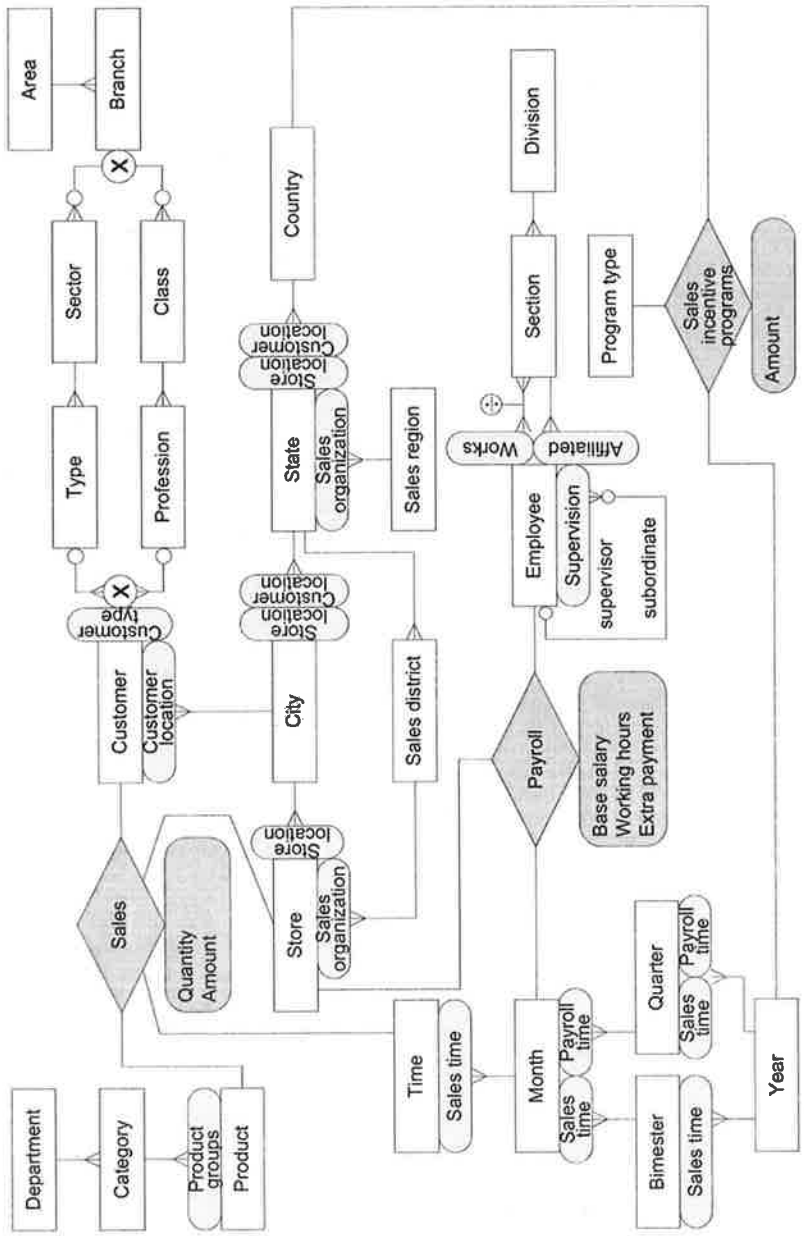
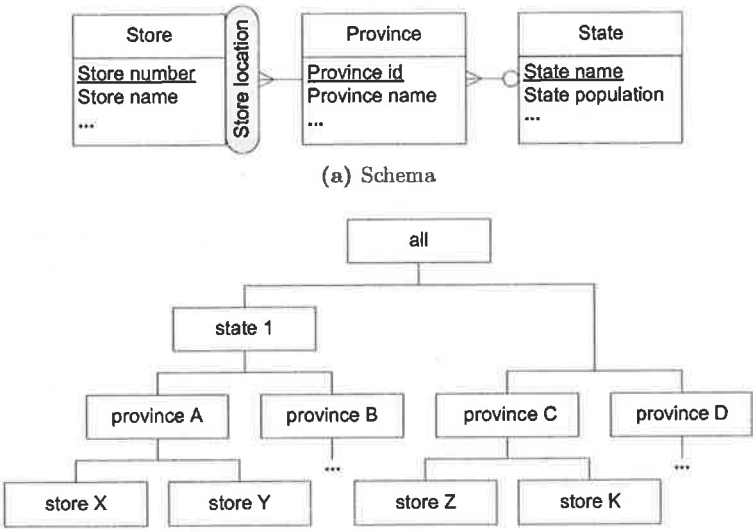


Fig. 3.20. A multidimensional schema containing several kinds of hierarchies



(a) Schema

(b) Examples of instances

Fig. 3.21. A generalized hierarchy with different root levels

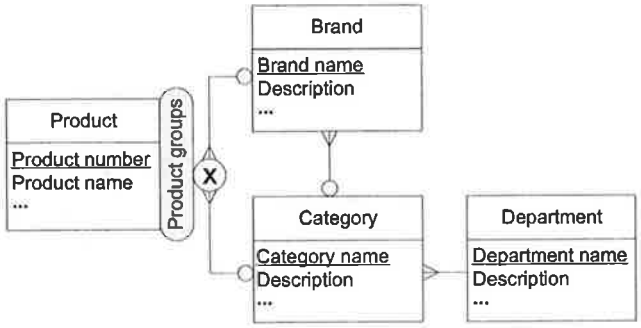


Fig. 3.22. A generalized hierarchy with three aggregation paths and different root levels

3.3.2 Role-Playing Dimensions

In the MultiDim model, a fact relationship represents an n-ary relationship between leaf levels. A leaf member participates in a fact relationship zero to many times. For example, in Fig. 3.20, if some product hast not (yet) been

sold, it will not be referenced in the Sales fact relationship; on the other hand, if a product has been sold in different stores or at different dates, it will participate several times in the fact relationship, each time with corresponding values of the Quantity and Amount measures.

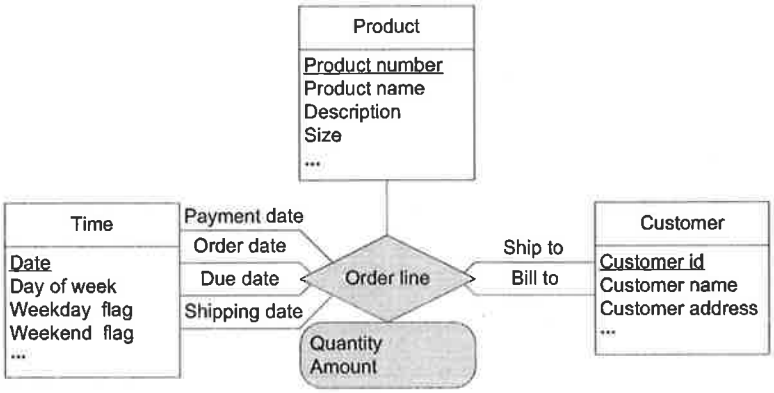


Fig. 3.23. Example of a role-playing dimension

As already mentioned at the beginning of this chapter in Sect. 3.1, in some situations a level may play several different roles in a fact relationship [101, 170]. To indicate these different roles, we include in the schema additional links between the leaf level and the fact relationship, each with the corresponding role name. The example shown in Fig. 3.23 may be used for analyzing the time when a customer orders a product (Order date), when the product is shipped (Shipping date), when the product should be delivered (Due date), and when it is actually paid for (Payment date). Furthermore, since a product can be shipped to one customer and be paid for by another, the Customer dimension also plays two different roles, i.e., Ship to and Bill to.

The schema in Fig. 3.23 could include four distinct time dimensions indicating the order, shipping, due, and payment dates, and two distinct dimensions referring to the clients to whom the products are shipped and the clients who pay for them. However, as can be seen in the figure, the complexity of the schema is reduced if the different roles that can be played by the dimensions are used instead of duplicating the dimensions.

Note that role-playing dimensions are supported by Microsoft Analysis Services 2005 [101, 157].

3.3.3 Fact Dimensions

We refer now to **fact dimensions** [157], or **degenerate dimensions** [147, 170]. A typical example is shown in Fig. 3.24, where users need to analyze product sales at the lowest granularity, which is represented by the transaction (or order) line. Note that fact relationship in Fig. 3.24 differs from the fact relationship Sales in Fig. 3.20, since in the latter the daily sales of products are summarized prior to their inclusion in the fact relationship.

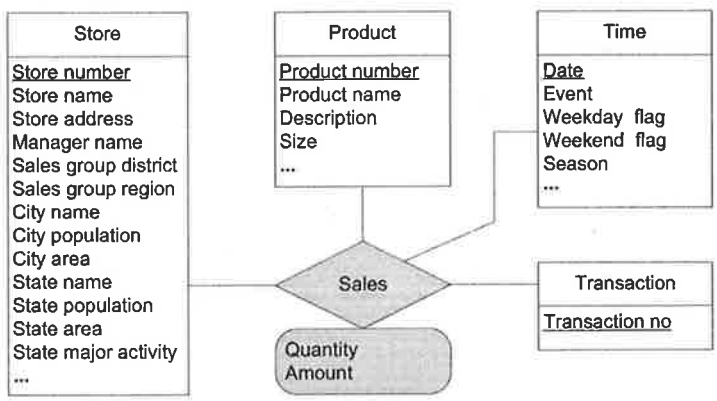


Fig. 3.24. A schema containing a fact dimension

As can be seen in Fig. 3.24, the Transaction dimension includes only the transaction number, since other data, such as the transaction date or the store number, are already included in the other dimensions. In logical representations of this schema, this dimension is typically converted into an attribute of the fact relationship. As a consequence, this dimension is called either a degenerate dimension, since it does not have other attributes describing it, or a fact dimension, since it is included as part of the fact relationship while playing the role of a dimension, i.e., it is used for grouping purposes.

We consider that including the dimension attribute in the fact relationship must be done at the logical or implementation level. The conceptual schema should keep this dimension to indicate that users may require grouping according to dimension members, i.e., according to the transaction number.

3.3.4 Multivalued Dimensions

The terms "multivalued dimensions" [147] and "many-to-many relationships between facts and dimensions" [170, 233, 283] have been used for representing the situation where several members of a dimension participate in the same

instance of a fact relationship. A common example used in this case is an analysis of clients' balances in bank accounts, as shown in Fig. 3.25.

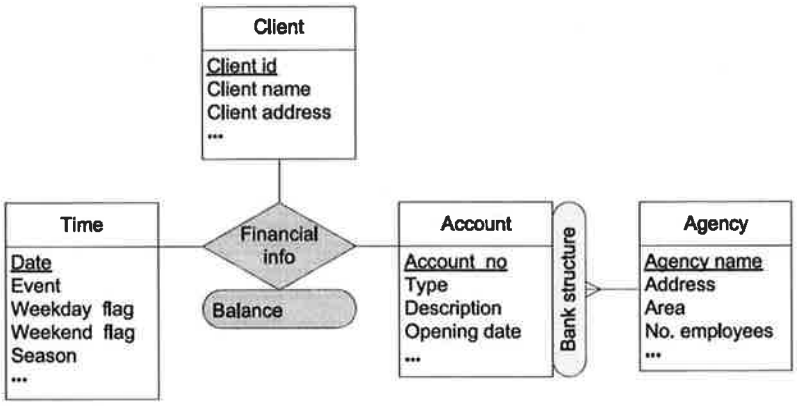


Fig. 3.25. Multidimensional schema for analysis of bank accounts

Since an account can be jointly owned by several clients, aggregation of the balance according to the clients will count this balance as many times as the number of account holders. Let us consider the example in Fig. 3.26. At some point in time T1, we have two accounts A1 and A2 with balances of, respectively, 100 and 500. As shown in the figure, both accounts are shared between several clients: account A1 between C1, C2, and C3, and account A2 between C1 and C2. The total balance of the two accounts is equal to 600; however, aggregation (for example, according to the Time or the Client dimension) gives a value equal to 1300.

Time	Account	Client	Balance
T1	A1	C1	100
T1	A1	C2	100
T1	A1	C3	100
T1	A2	C1	500
T1	A2	C2	500

Fig. 3.26. Example of double-counting problem for a multivalued dimension

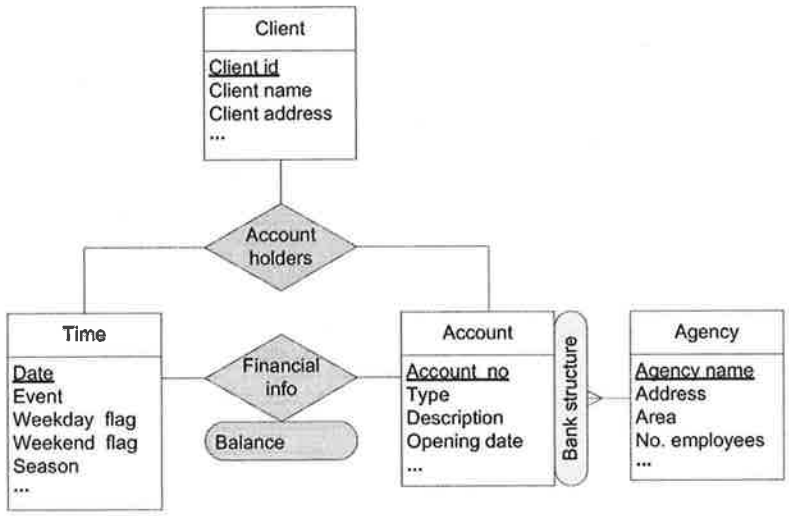
The problem of **double counting** can be analyzed by considering the **multidimensional normal forms** (MNFs) defined by Lehner et al. [163] and Lechtenbörger and Vossen [161]. MNFs focus on summarizability (see

Sect. 2.6.2) and determine conditions that ensure correct measure aggregation in the presence of generalized hierarchies (see Sect. 3.2.1). MNFs are defined using a generalization of functional dependencies that takes null values into account. The first multidimensional normal form (1MNF) establishes four conditions. The first three conditions aim at ensuring that the functional dependencies implied by a schema reflect those in the application domain. The last condition requires that each measure is uniquely identified by the set of leaf levels. The 1MNF is the basis for correct schema design and is used to define the remaining MNFs. The second and third multidimensional normal forms focus on establishing conditions for generalized hierarchies. We omit their descriptions here since our model is more general than that of Lechtenbörger and Vossen [161], i.e., the MultiDim model includes nonstrict (Sect. 3.2.2) and parallel dependent hierarchies (Sect. 3.2.4) and also allows one to share levels among dimensions (as in Fig. 3.20). In addition, the MNFs do not take multivalued dependencies into account, and these dependencies are important to ensure correctness of multidimensional schemas.

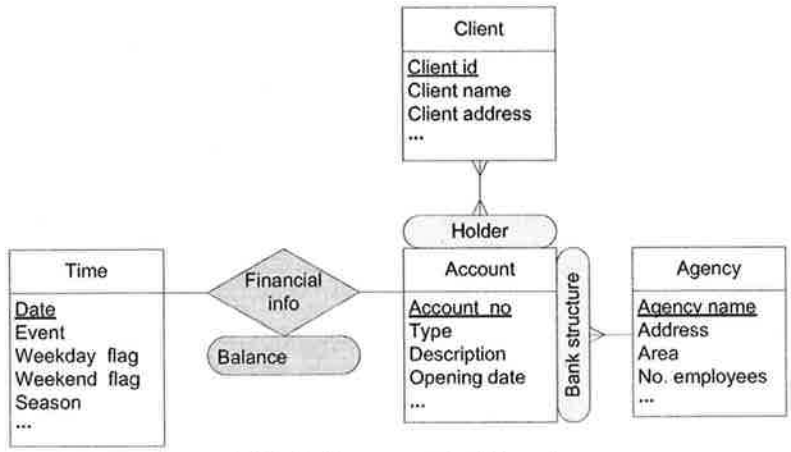
Let us analyze the schema in Fig. 3.25 in terms of the 1MNF: we need to establish the functional dependencies that exist between the leaf levels and the measures. Since the balance depends on the specific account and the time when it is considered, we have the situation that the account and the time determine the balance. Therefore, the schema in Fig. 3.25 does not satisfy the 1MNF, since the measure is not determined by all leaf levels, and thus the fact relationship must be decomposed.

If we consider the multivalued dependencies, there are two possible ways in which the Financial info fact relationship in Fig. 3.25 can be decomposed. In the first case, the same joint account may have different clients assigned to it during different periods of time, and thus the time and the account multidetermine the clients. This situation leads to the solution shown in Fig. 3.27a, where the original fact relationship is decomposed into two fact relationships, i.e., Account holders and Financial info. On the other hand, the second case arises when the client composition of joint accounts does not change over time, and thus only the account multidetermines the clients. In this case, the link relating in Fig. 3.27a the Time level and the Account holders fact relationship should be eliminated. Alternatively, this situation can be modeled with a nonstrict hierarchy as shown in Fig. 3.27b.

Even though the solutions proposed in Fig. 3.27 eliminate the double-counting problem, the two schemas in Fig. 3.27 require programming effort for queries that ask for information about individual clients. The difference lies in the fact that in Fig. 3.27a a drill-across operation (see Sect. 2.6.3) between the two fact relationships is needed, while in Fig. 3.27b special procedures for aggregation in nonstrict hierarchies must be applied. In the case of Fig. 3.27a, since the two fact relationships represent different granularities, queries with drill-across operations are complex, demanding a conversion either from a finer to a coarser granularity (for example, grouping clients to know who holds a specific balance in an account) or vice versa (for example, distributing a



(a) Creating two fact relationships



(b) Including a nonstrict hierarchy

Fig. 3.27. Two possible decompositions of the fact relationship in Fig. 3.25

balance between different account holders). Note also that the two schemas in Fig. 3.27 could represent the information about the percentage of ownership of accounts by customers (if this is known). This could be represented by a measure in the Account holders fact relationship in Fig. 3.27a, and by a distributing factor in the many-to-many relationship in Fig. 3.27b.

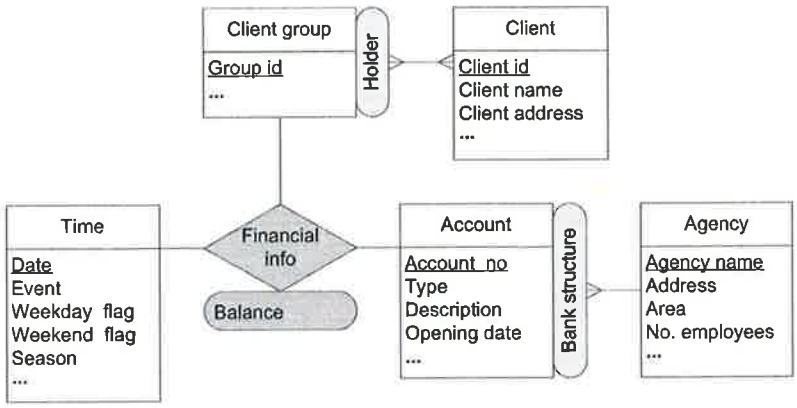


Fig. 3.28. Alternative decomposition of the schema shown in Fig. 3.25

Another solution to this problem, implicitly presented by Pedersen [232] and Song et al. [283], is shown in Fig. 3.28. In this solution, an additional level is created, which represents the groups of clients participating in joint accounts. For our example in Fig. 3.26, we create two groups: one that includes clients C1, C2, and C3, and another with clients C1 and C2. The difference between Fig. 3.28 and Pedersen's solution consists in the fact that the latter includes the groups of clients together with individual clients, i.e., the same level will contain members with different data granularities. However, the schema in Fig. 3.28 is not in the 1MNF, since the measure Balance is not determined by all leaf levels, i.e., it is only determined by Time and Account. Therefore, the schema must be decomposed leading to schemas similar to those in Fig. 3.27, with the difference that in this case the Client level in the two schemas in Fig. 3.27 is replaced by a nonstrict hierarchy composed of the Client group and the Client levels.

Another solution for avoiding multivalued dimensions is to choose one client as the primary account owner and ignore the other clients. In this way, only one client will be related to a specific balance, and the schema in Fig. 3.25 can be used without any problems related to double counting of measures. However, this solution may not represent the real-world situation, and may exclude from the analysis the other clients of joint accounts.

In summary, we can avoid multivalued dimensions in multidimensional schemas by using one of the solutions presented in Fig. 3.27. The choice between these alternatives depends on the functional and multivalued dependencies existing in the fact relationship, the kinds of hierarchies that are allowed in the schema, and the complexity of the implementation.

3.4 Metamodel of the MultiDim Model

In this section, we present the metamodel of the MultiDim model using UML notation, as shown in Fig. 3.29.

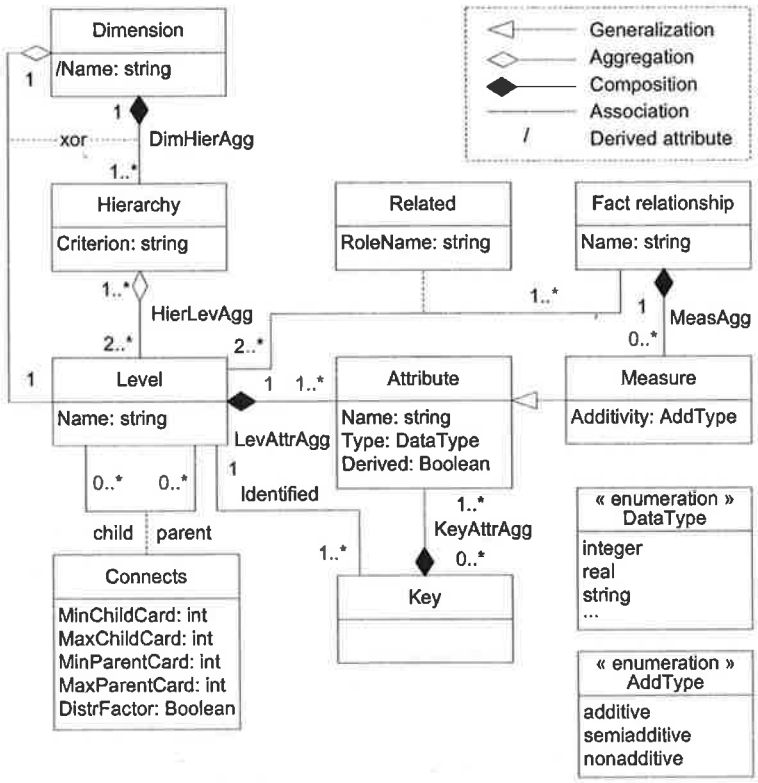


Fig. 3.29. Metamodel of the MultiDim model

A dimension is composed of either one level or one or more hierarchies, and each hierarchy belongs to only one dimension. A hierarchy contains two or more related levels that can be shared between different hierarchies. A criterion name identifies a hierarchy and each level has a unique name. The leaf-level name is used to derive a dimension's name: this is represented by the derived attribute Name in Dimension. Levels include attributes, some of which are key attributes used for aggregation purposes, while others are descriptive attributes. Attributes have a name and a type, which must be a data type, i.e., integer, real, string, etc. Further, an attribute may be derived.

The levels forming hierarchies are related through the Connects association class. This association class is characterized by the minimum and maximum cardinalities of the parent and child roles, and the distributing factor, if any. A constraint not shown in the diagram specifies that a parent-child relationship has a distributing factor only if the maximum cardinalities of the child and the parent are equal to "many".

A fact relationship represents an n-ary association between leaf levels with $n > 1$. Since leaf levels can play different roles in this association, the role name is included in the Related association class. A fact relationship may contain attributes, which are commonly called measures; they may be additive, semiadditive, or nonadditive.

3.5 Mapping to the Relational and Object-Relational Models

3.5.1 Rationale

Conceptual models can be implemented using various logical models. In this section, we present the mapping rules that allow one to implement conceptual schemas designed using the MultiDim model into the relational and object-relational models.

As already explained in Sect. 2.7, data warehouses are usually implemented in relational databases as star or snowflake schemas. There are several reasons for using a relational implementation for multidimensional models.

- Relational databases use well-known strategies for data storage, and are well standardized, tool independent, and readily available. In contrast, there is no consensus in the research community about defining a logical level for OLAP systems [8]. Furthermore, multidimensional OLAP (MOLAP) systems differ greatly in the structures used for data storage.
- Much research has been done in the field of relational databases on improving query optimization, indexing, join operations, and view materialization, considering the particularities of data warehouse applications.
- Even though MOLAP systems offer better performance during roll-up operations, many commercial and prototype systems use ROLAP (relational OLAP) or HOLAP (Hybrid OLAP, a combination of ROLAP and MOLAP) storage for large volumes of data, as described in Sect. 2.7.
- Commercial relational database systems [14, 101, 216] include extensions to represent and manage a multidimensional view of data.
- Relational databases are one of the data resources in the **Common Warehouse Model** (CWM) [209] from the Object Management Group (OMG).[9]

[9] See the description of the CWM in Sect. 3.8.

The object-relational model is increasingly being used in conventional database applications. One reason for this is that it preserves the foundations of the relational model while organizing data using an object model [155] (see Sect. 2.3.2). In addition, the object-relational model is supported by many commercial DBMSs. Since data warehouses are specific databases devoted to analytical purposes, they can also be implemented using the object-relational model.

We chose as implementation models, the surrogate-based relational model [90] and the the object-relational model. In these models, each row in a table has an associated surrogate, i.e., a system-generated artificial key. Surrogate values cannot be seen or modified by users. The reason for using surrogates instead of relying on user-defined keys is that data warehouses usually use system-generated keys to ensure both better performance during join operations and independence from transactional systems. Further, surrogates do not vary over time, so that two entities that have identical surrogates represent the same entity, thus allowing one to include historical data in an unambiguous way. In addition, in some situations it is necessary to store information about an entity either before it has been assigned a user-controlled key value (for example, to introduce a student into a database before an ID number has been assigned to him/her) or after it has ceased to have one (for example, when an employee leaves an organization). Further, the SQL:2003 standard [194, 195] and some current DBMSs (for example, Oracle [215]) support tables with rows that have automatically created surrogates. Nevertheless, the relational or object-relational models without surrogates can also be used in our mapping for those DBMSs that do not allow surrogates. In this case, implementers are responsible for creating artificial keys independent from those used in the source systems.

3.5.2 Mapping Rules

Since the MultiDim model is based on the ER model, its mapping to the relational model is based on well-known rules, such as those described in Sect. 2.3.1:

Rule 1: A level corresponds to an entity type in the ER model. It is mapped to a table that contains all attributes of the level and includes an additional attribute for a surrogate key.

Rule 2: A parent-child relationship corresponds to a binary relationship type in the ER model. Recall that in order to have meaningful hierarchies, the maximum cardinality of the parent role must be n. Thus, two different mappings exist, depending on the cardinality of the child role:

Rule 2a: If the cardinality of the child role is (0,1) or (1,1), i.e., the parent-child relationship is one-to-many, the table corresponding to the child level is extended with the surrogate key of the corresponding

parent level, i.e., there is a foreign key in the child table pointing to its parent table.

Rule 2b: If the cardinality of the child role is (0,n) or (1,n), i.e., the parent-child relationship is many-to-many, a new table is created that contains as attributes the surrogate keys of the parent and child levels. If the parent-child relationship has a distributing factor, an additional attribute is added to the table to store this information.

Rule 3: A fact relationship type corresponds to an n-ary relationship type in the ER model. A fact relationship involving leaf levels $L_1, \ldots, L_n$ is mapped to a table that includes as attributes the surrogate keys of the participating levels, one for each role that each level is playing. In addition, every attribute (i.e., measure) of the fact relationship is mapped to an attribute of the table.

Applying the above rules to the example given in Fig. 3.2 yields the tables shown in Fig. 3.30. The Sales table includes five foreign keys, i.e., one for each participating level, and three attributes representing the measures, i.e., Quantity, Price, and Amount. Note that when a level plays several roles, as is the case for Time, each role is represented by a foreign key, i.e., Order date and Payment date are both foreign keys to the Time dimension. Note also that the many-to-many parent-child relationship between Product and Distributor is mapped to the table ProdDist, containing two foreign keys.

A set of SQL DDL commands defining an excerpt from the relational schema of Fig. 3.30 is as follows.

```
create table Sales as (
    ProductFkey integer not null,
    OrderTimeFkey integer not null,
    PaymentTimeFkey integer not null,
    StoreFkey integer not null,
    CustomerFkey integer not null,
    Quantity integer not null,
    Price decimal(7,2) not null,
    Amount decimal(7,2) not null,
    primary key (ProductFkey, OrderTimeFkey, PaymentTimeFkey,
        StoreFkey, CustomerFkey),
    foreign key ProductFkey references Product(ProductKey),
    foreign key OrderTimeFkey references Time(TimeKey),
    foreign key PaymentTimeFkey references Time(TimeKey),
    foreign key StoreFkey references Store(StoreKey),
    foreign key CustomerFkey references Customer(CustomerKey) );
create table Product as (
    ProductKey integer primary key not null,
    ProductNumber character varying(9) not null,
    ProductName character varying(30) not null,
    Description text not null,
    Size integer not null,
```

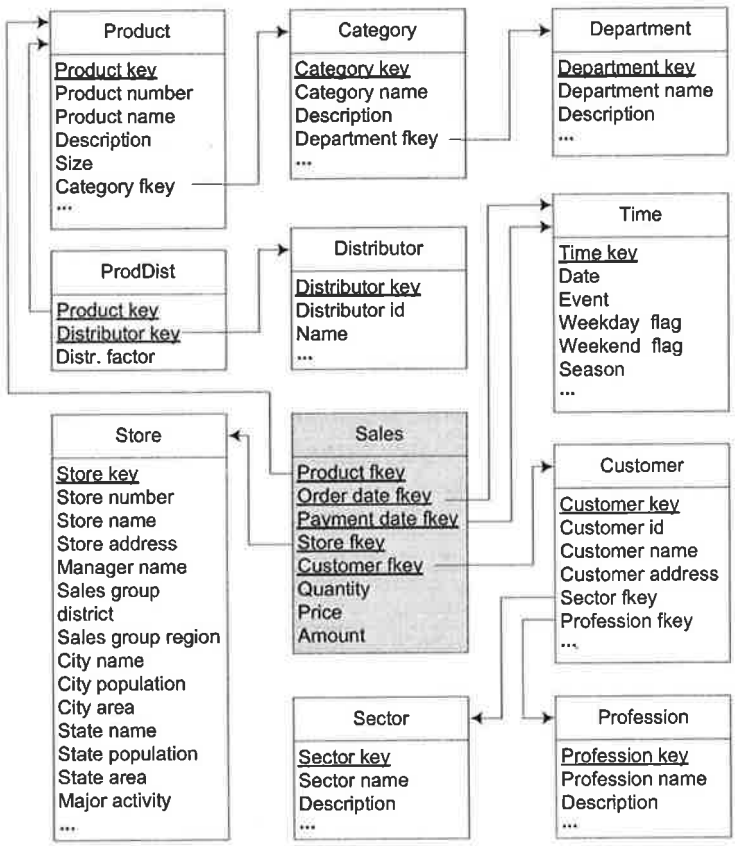


Fig. 3.30. Relational representation of the example in Fig. 3.2

```
        CategoryFkey integer not null,
        ...
        foreign key CategoryFkey references Product(ProductKey) );
create table Category as (
        CategoryKey integer primary key not null,
        CategoryName character varying(20) not null,
        Description text not null,
        DepartmentFkey integer not null,
        ...
        foreign key DepartmentFkey references Department(DepartmentKey) );
create table Department as (
        DepartmentKey integer primary key,
        DepartmentName character varying (30) not null,
```

```
        Description text not null,
        ... );
```

In the small excerpt given above, we have defined only the Sales fact table and the tables for the hierarchy composed of the levels Product–Category–Department. The definitions for the other tables are similar.

The transformation of the MultiDim model to the object-relational model is based on rules similar to those specified above for the mapping to the relational model.[10] The difference is that, as explained in Sect. 2.3.2, the object-relational model provides a ref type, which can be used instead of foreign keys in the above rules.

The set of SQL:2003 commands defining an object-relational schema corresponding to the relational schema of Fig. 3.30 is as follows.

```
create type SalesType as (
    ProductRef ref(ProductType) scope Product references are checked,
    OrderTimeRef ref(TimeType) scope Time references are checked,
    PaymentTimeRef ref(TimeType) scope Time references are checked,
    StoreRef ref(StoreType) scope Store references are checked,
    CustomerRef ref(CustomerType) scope Customer references are checked,
    Quantity integer,
    Price decimal(7,2),
    Amount decimal(7,2) )
    ref is system generated;
create table Sales of SalesType (
    constraint SalesPK primary key (ProductRef, OrderTimeRef,
        PaymentTimeRef, StoreRef, CustomerRef),
    ProductRef with options constraint ProductRefNN ProductRef not null,
    /* not null constraints for all other attributes */
    ref is oid system generated );
create type ProductType as (
    ProductKey integer,
    ProductNumber character varying(9),
    ProductName character varying(30),
    Description text,
    Size integer,
    CategoryRef ref(CategoryType) scope Category references are checked,
    ... )
    ref is system generated;
create table Product of ProductType (
    constraint ProductPK primary key (ProductKey),
    ref is oid system generated );
create type CategoryType as (
    CategoryKey integer,
    CategoryName character varying(20),
    Description text,
```

[10] As explained in Sect. 3.6.2, there is one exception to this, since the relational and the object-relational representation of nonstrict hierarchies may differ.

```
        DepartmentRef ref(DepartmentType) scope Department
            references are checked,
        ... )
        ref is system generated;
    create table Category of CategoryType (
        constraint CategoryPK primary key (CategoryKey),
        ref is oid system generated );
    create type DepartmentType as (
        DepartmentKey integer,
        DepartmentName character varying (30),
        Description text,
        ... )
        ref is system generated;
    create table Department of DepartmentType (
        constraint DepartmentPK primary key (DepartmentKey),
        ref is oid system generated );
```

Note that we have implemented the above schema using typed tables. As already said above, typed tables allow the system-generated identifiers to be used for linking rows between tables. Typed tables also allow methods to be defined for the types associated with the tables. However, since we do not refer to any aspects of manipulation in the following, we have not included the specification of methods for object-relational schemas. Note also that constraints associated with the types (for instance, primary key constraints) are specified in the corresponding table declarations. For brevity, in the above SQL commands we have not specified many constraints, in particular not null constraints for mandatory attributes. An example of such a constraint is given for the attribute ProductRef in the Sales table.

3.6 Logical Representation of Hierarchies

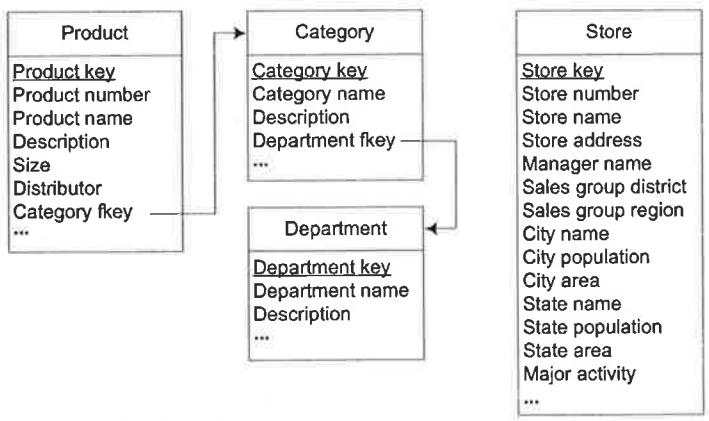
The general mapping rules given in the previous section do not capture the specific semantics of all of the kinds of hierarchies described in Sect. 3.2. In addition, for some kinds of hierarchies, alternative logical representations exist. In this section, we consider in detail the logical representation of the various kinds of hierarchies. In what follows we discuss the relational representation, and we mention the object-relational representation when it differs from the relational one.

3.6.1 Simple Hierarchies

Balanced Hierarchies

Applying the mapping rules described in Sect. 3.5.2 to balanced hierarchies leads to the well-known relational representations of data warehouses:

- **Normalized tables** or **snowflake structure:** each level is represented as a separate table that includes a key and the descriptive attributes of the level. For example, using Rules 1 and 2a of Sect. 3.5.2 for the Product groups hierarchy in Fig. 3.4 gives a snowflake structure with tables Product, Category, and Department, as shown in Fig. 3.31a.
- **Denormalized** or **flat tables:** the key and descriptive attributes of all levels forming a hierarchy are included in one table. This structure can be obtained in two ways: (1) denormalizing the tables that represent several hierarchy levels (for example, including in one table all attributes of the Product, Category, and Department tables shown in Fig. 3.31a), or (2) mapping a dimension that includes a one-level hierarchy according to Rule 1 (for example, the Store dimension in Fig. 3.2 may be represented as shown in Fig. 3.31b).



(a) Snowflake structure **(b)** Flat table

Fig. 3.31. Relations for a balanced hierarchy

Normalized tables are used in **snowflake schemas** (see Sect. 2.7). They represent hierarchical structures better, since every level can be easily distinguished and, further, levels can be reused between different hierarchies. Additionally, this representation can easily manage heterogeneity across levels [130], i.e., it allows different levels of a hierarchy to include specific attributes. For example, the Product, Category, and Department tables in Fig. 3.31a have specific attributes. This data structure allows measures to be aggregated using, for example, the SQL group by, roll-up, or cube operators (see Sect. 2.6.3). Further, in some applications, snowflake schemas can improve system performance in spite of requiring join operations between the relations representing

the levels [190]. However, snowflake schemas have some limitations; for example, they cannot be used without modification if there is heterogeneity within a level [130], i.e., different characteristics within a group of members of a level. This situation may occur when different products have their own specific attributes, as in the case of generalized hierarchies (see Sect. 3.2.1).

On the other hand, the inclusion of flat tables in the schema, leading to **star schemas** (see Sect. 2.7), has several advantages: the schema is easy to understand for the purpose of query formulation, it can be easily accessed, and few joins are needed for expressing queries, owing to the high level of denormalization. Additionally, much research has been done to improve system performance during processing of star queries (e.g., [83, 142]). Finally, as for the normalized approach, the aggregations of measures are straightforward.

Some disadvantages of the star schema representation are as follows: it does not allow hierarchies to be modeled adequately, since the hierarchy structure is not clear and additional information is required to represent it [130]. For example, for the Store dimension in Fig. 3.31b, it is not clear which attributes can be used for hierarchies. As can also be seen in the figure, it is difficult to clearly associate attributes within their corresponding levels. Moreover, in hierarchies that have many levels, the number of attributes is at least as large as the hierarchy depth, making the hierarchy structure difficult to understand. Owing to denormalization, and consequently data redundancy, other well-known problems also arise.

Unbalanced Hierarchies

Since unbalanced hierarchies do not satisfy the summarizability conditions [164] (see Sect. 2.6.2), the mapping described in Sect. 3.5.2 may lead to the problem of excluding from the analysis the members of higher levels that do not have child members. For instance, since in Fig. 3.5a all measures are associated with the leaf level, i.e., the ATM level, these measures will be aggregated into the higher levels only for those agencies that have ATMs and, similarly, only for those branches that have agencies. To avoid this problem, an unbalanced hierarchy can be transformed into a balanced one. This transformation is performed using placeholders (marked PH in Fig. 3.32) [233] or null values [157] in missing levels. Afterwards, the logical mapping for balanced hierarchies may be applied.

Nevertheless, this representation has the following shortcomings:

1. A fact table must include common measures belonging to different hierarchy levels (in our example, ATMs, agencies, and branches).
2. The common measures used for all levels have different data granularities. For example, measures for the ATM level and for the Agency level for those members that do not have an ATM are included in the same fact table.
3. Special placeholders must be created and managed for aggregation purposes. For example, in Fig. 3.32 the same measure value must be repeated for branch 2, while using two placeholders for the missing levels.

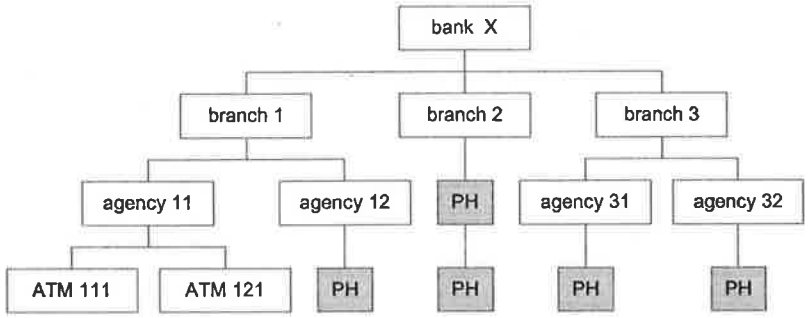


Fig. 3.32. Transformation of the unbalanced hierarchy shown in Fig. 3.5b into a balanced one using placeholders

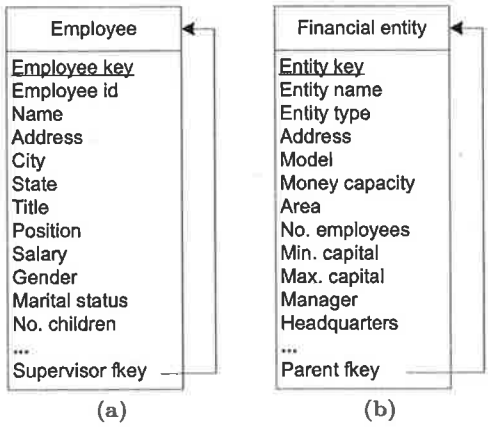


Fig. 3.33. Relational implementation of the recursive hierarchies shown in Fig. 3.6

4. The introduction of meaningless values requires more storage space.
5. A special interface must be developed to hide placeholders from users.

Recall from Sect. 3.2.1 that **recursive hierarchies** are a special case of unbalanced hierarchies. Mapping recursive hierarchies to the relational model yields **parent-child tables** containing all attributes of a level, and an additional foreign key relating child members to their corresponding parent. Figure 3.33a shows the table representing the recursive hierarchy shown in Fig. 3.6a.

Although a parent-child table allows one to represent the semantics of recursive hierarchies, operations over it are more complex. In particular, **recursive queries** are necessary for traversing the levels forming a recursive

hierarchy. Recursive queries are included in the SQL:2003 standard [194] and are supported by several commercial systems, such as SQL Server, Oracle, and DB2.

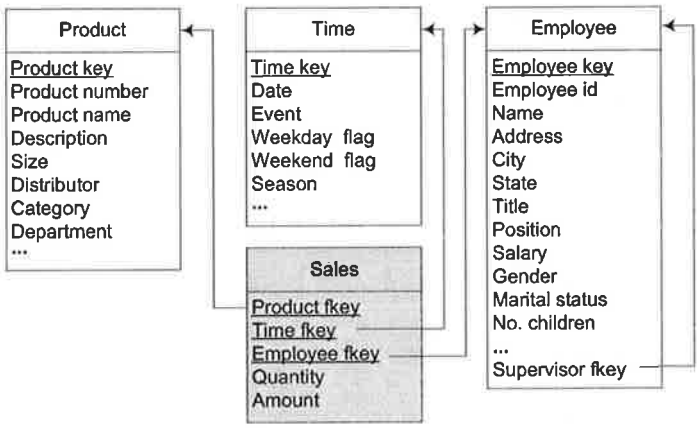


Fig. 3.34. A schema with a recursive hierarchy in the Employee dimension

Consider the example depicted in Fig. 3.34 of a data warehouse for analyzing sales by employees in the case of direct marketing, i.e., when the goods are sold directly to the customer. Suppose now that the user is interested in finding, for each supervisor, the total amount of sales made by his/her subordinates during 2006. This requires a recursive query in SQL as follows:

```
with Supervision(SupervisorKey, SubordinateKey) as
(     select SupervisorFKey, EmployeeKey
      from Employee
      where SupervisorFKey is not null
union all
      select S.SupervisorKey , E.EmployeeKey
      from Supervision S, Employee E
      where S.SubordinateKey = E.SupervisorFKey
)
select SupervisorKey, sum(S1.Amount)
from Sales S, Supervision U, Time T
where S.EmployeeFKey=U.SubordinateKey
and S1.TimeFkey=T.TimeKey
and T.Date >= '1/1/2006' and T.Date =< '31/12/2006'
group by SupervisorKey
```

In the above query, the clause with defines a temporary table Supervision that computes recursively all direct and indirect subordinates of an employee. For example, if E1 supervises E2, and E2 supervises E3, the Supervision table will

contain in addition one line stating that E1 supervises E3. Then, the select statement joins the tables Supervision, Sales, and Time and groups the result by SupervisorKey in order to sum all sales of the subordinates.

Financial entity											
Entity key	Entity name	Entity type	Addr.	Model	Money capacity	Area	No. emp.	Min. capital	Max. capital	...	Parent fkey
1	ATM111	ATM	...	T1	100000	null	null	null	null	...	3
2	ATM112	ATM	...	T2	150000	null	null	null	null	...	3
3	agency11	Agency	...	null	null	150	12	null	null	...	5
4	agency12	Agency	...	null	null	135	9	null	null	...	5
5	branch1	Branch	...	null	null	null	null	10000	99000	...	10
6	branch2	Branch	...	null	null	null	null	1500	55000	...	10
7	agency31	Agency	...	null	null	230	15	null	null	...	9
8	agency32	Agency	...	null	null	185	12	null	null	...	9
9	branch3	Branch	...	null	null	null	null	7500	98500	...	10
10	bankX	Bank	...	null	null	null	null	null	null	...	null

Fig. 3.35. Parent-child relational schema of an unbalanced hierarchy with instances from Fig. 3.5b

One solution for implementing unbalanced hierarchies is to transform them into recursive hierarchies, which are then transformed at the logical level into **parent-child tables**. For example, such a transformation of the recursive hierarchy in Fig. 3.6b will give the table shown in Fig. 3.33b. However, the problem of such tables is that they represent semantically different levels, each one having particular attributes. As can be seen in Fig. 3.35, since a parent-child table contains the attributes of all hierarchy levels, a member of a level will contain null values for all attributes belonging to the other levels.

For this reason, **integrity constraints** must be implemented to ensure that a member has values for the attributes of its level. For example, the following check constraint states that members of the ATM level must have values for the attributes Address, Model, and Money capacity and null values for the other attributes:

```
alter table FinancialEntity
    add constraint ATMAttributes
    check ( (EntityType != 'ATM') or ( Address is not null and
    Model is not null and MoneyCapacity is not null and
    Area is null and NoEmployees is null and ... ) )
```

Note that in the above constraint the logical implication (EntityType = 'ATM') $\Rightarrow$ (...) is encoded as (EntityType != 'ATM') $\vee$ (...). Similar check constraints are needed for the other levels of the hierarchy, i.e., for the Agency, Bank, and Branch levels.

Other triggers are needed to ensure that the links between levels are correct. For example, the following trigger, implemented in Microsoft SQL Server, enforces the constraint that ATM members are related to Agency members:

```
create trigger ATM_Agency on FinancialEntity
after insert, update as
if exists ( select * from Inserted I, FinancialEntity F
      where I.ParentFkey = F.EntityKey and
      I.EntityType = 'ATM' and F.EntityType != 'Agency' )
begin
      raiserror 13000 'An ATM member must be related to an Agency member'
end
```

Generalized Hierarchies

Several approaches have been proposed for the logical representation of generalized hierarchies: (1) representing heterogeneity within levels by creating separate tables for each level of the hierarchy and keeping the information about which tables compose the different paths [130]; (2) using a flat representation with null values for attributes that do not describe specific members [163] (for example, companies will have null values in attributes corresponding to persons); (3) using separate star schemas, i.e., separate fact and dimension tables for each path [16, 95, 147]; and (4) creating one table for the common levels and another table for the specific ones [16, 275].

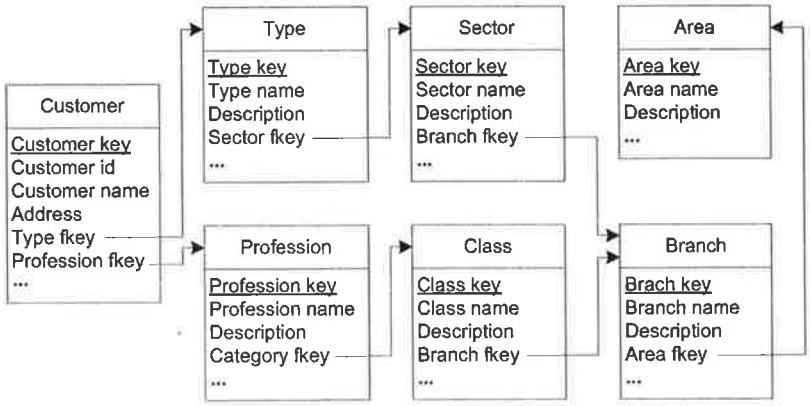


Fig. 3.36. Relations for the generalized hierarchy shown in Fig. 3.8

One disadvantage of the first three approaches above is that the common levels of the hierarchy cannot be easily distinguished and managed. Further,

the inclusion of null values requires one to specify additional constraints to ensure correct queries (for example, to avoid grouping Type with Profession in Fig. 3.8). Even though these difficulties do not exist for solution (4) above, in that case an additional attribute must be created in the table representing the common levels of the hierarchy to specify the name of the table for the specific levels. In consequence, queries that manage metadata and data are required to access the tables of the specific levels, which is not an easy task to accomplish in SQL. The traditional mapping of generalization/specialization to relational tables (e.g., Rule 7 in Sect. 2.3.1) also gives problems owing to the inclusion of null values and the loss of the hierarchical structure.

Applying the mapping described in Sect. 3.5.2 to the generalized hierarchy given in Fig. 3.8 yields the relations shown in Fig. 3.36. Even though this mapping clearly represents the hierarchical structure, it does not allow one to traverse only the common levels of the hierarchy. To ensure heterogeneity within levels and the possibility of traversing either specific or common levels, we propose in addition the following mapping rule:

Rule 4: A table corresponding to a splitting level in a generalized hierarchy may have an additional attribute, which is a foreign key of the next joining level, provided this level exists. The table may also include a discriminating attribute that indicates the specific aggregation path of each member.

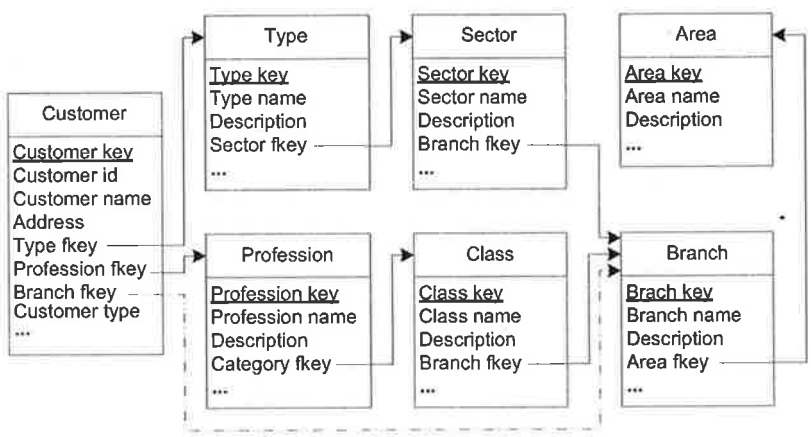


Fig. 3.37. Improved relational representation for the generalized hierarchy shown in Fig. 3.8

An example of the relations for the hierarchy in Fig. 3.8 is given in Fig. 3.37. The table Customer includes two kinds of foreign keys: one that

indicates the next specialized hierarchy level (for example, Profession fkey), which is obtained by applying Rules 1 and 2a in Sect. 3.5.2, and another one corresponding to the next joining level (for example, Branch fkey), which is obtained by applying the above Rule 4. The discriminating attribute Customer type, with values Person and Company, indicates the specific aggregation path of members to facilitate aggregations. Finally, integrity constraints must be specified to ensure that only one of the foreign keys for the specialized levels may have a value, according to the value of the discriminating attribute:

```
alter table Customer
    add constraint CustomerTypeCK
    check ( CustomerType in ('Person', 'Company') )
alter table Customer
    add constraint CustomerPersonFK
    check ( (CustomerType != 'Person') or
    ( ProfessionFkey is not null and TypeFkey is null ) )
alter table Customer
    add constraint CustomerCompanyFK
    check ( (CustomerType != 'Company') or
    ( ProfessionFkey is null and TypeFkey is not null ) )
```

The structure in Fig. 3.37 allows one to choose alternative paths for analysis. One possibility is to use paths including the specific levels, for example Profession or Type. Another possibility is to access the levels that are common to all members, i.e., to ignore the levels between the splitting and joining levels; for example, one may analyze all customers, whatever their type, using the hierarchy Customer–Branch–Area.

As with the snowflake structure, one disadvantage of this structure is the necessity to apply join operations between several tables. However, an important advantage is the expansion of the analysis possibilities that it offers by taking into account the specific and general characteristics of hierarchy members.

As a **noncovering hierarchy** is a special case of a generalized hierarchy, the mapping proposed above can also be applied for such hierarchies. Nevertheless, since in a noncovering hierarchy there is a unique path, where only some levels are skipped, the usual solution (e.g., [101, 233]) is to transform the hierarchy by including placeholders in the missing intermediate levels. In this way, a noncovering hierarchy is converted into a balanced hierarchy and a star or snowflake structure can be used for its logical representation.

3.6.2 Nonstrict Hierarchies

The traditional mapping of nonstrict hierarchies to the relational model, as specified in Sect. 3.5.2, creates relations for representing the levels and an additional relation for representing the many-to-many relationship between them. An example for the case of the hierarchy in Fig. 3.11a is given in Fig. 3.38. The

table representing the many-to-many relationship (EmplSection in the figure) is usually called a **bridge table** [147].

Note that if the parent-child relationship has a distributing factor, the bridge table will include an additional attribute for storing the values required for measure distribution.[11] For example, for an employee X, the table EmplSection could include three rows: (1) employee X, section 1, 30%; (2) employee X, section 2, 50%; (3) employee X, section 3, 20%. In order to aggregate measures correctly, a special aggregation procedure that uses this distributing factor must be implemented.

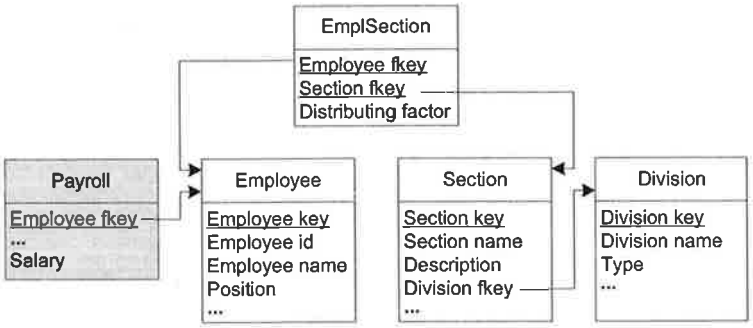


Fig. 3.38. Relational tables for the nonstrict hierarchy of Fig. 3.11

Recall from Sect. 3.2.2 that another solution is to transform a nonstrict hierarchy into a strict one by including an additional dimension in the fact relationship, as shown in Fig. 3.14. Then, the corresponding mapping for a strict hierarchy can be applied. The choice between the two solutions may depend on various factors, such as those mentioned in Table 3.1. Still another option is to convert the many-to-many relationship into a one-to-many relationship by defining a "primary" relationship [203], i.e., to convert the nonstrict hierarchy into a strict one, to which the corresponding mapping for a simple hierarchy is applied.

There are two possibilities for representing nonstrict hierarchies in the object-relational model. We can use a solution similar to the bridge table above, with the difference that references are used instead of foreign keys. The other solution extends the table representing a child level with an additional multivalued attribute (of an array or multiset type) that includes references to the parent members. If the schema contains a distributing factor, the additional multivalued attribute is composed of two attributes: one referencing a parent member and another one specifying the value of the distributing factor for this particular parent member. This is shown in Fig. 3.39.

[11] This is called the weighting factor in [147].

Table 3.1. Comparison of implementations of nonstrict hierarchies using a bridge table and an additional dimension

Factor	Bridge table	Additional dimension
Table size	The fact table is smaller.	The fact table is bigger if child members are related to many parent members. The additional foreign key in the fact table will also increase the space required.
Additional structure	Additional information about the parent-child relationship and distribution factor (if it exists) must be stored separately.	Measures are only included in the fact table.
Performance	Additional join operations, calculations, and programming effort are needed to aggregate measures correctly.	Measures in the fact table are ready for aggregation across the hierarchy.
Type of application	Useful for applications that have only a few nonstrict hierarchies, and other strict hierarchies referring to the same measures.	Implementation should be evaluated taking into account a space-time trade-off.
Changes with time	Useful when the information about measure distribution does not change with time.	Can easily represent changes in measure distribution that occur with time.
Current OLAP implementation	Current tools can manage to some extent this type of hierarchy.	Current tools can manage this type, but in some cases the problems of double counting may occur.

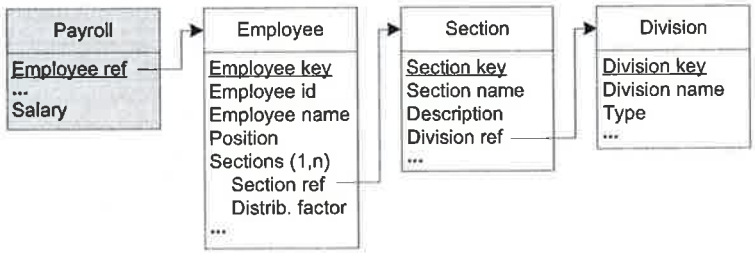


Fig. 3.39. Object-relational representation of the nonstrict hierarchy of Fig. 3.13

Even though nonstrict hierarchies are very common in real-world situations, not much research has been done to them. The search for solutions that allow better management of these hierarchies is considered to be of funda-

mental importance [104]. In this section, we have analyzed only the case of nonstrict balanced hierarchies. When one has a nonstrict hierarchy of one of the other types presented earlier, a combination of the proposed mapping for the specific hierarchy and the mapping for a nonstrict hierarchy should be applied.

3.6.3 Alternative Hierarchies

For alternative hierarchies, the traditional mapping to relational tables can be applied. This is shown in Fig. 3.40 for the conceptual schema in Fig. 3.16. Note that even though generalized and alternative hierarchies can be easily distinguished at the conceptual level (see Figs. 3.8a and 3.16), this distinction cannot be made at the logical level (compare Figs. 3.36 and 3.40).

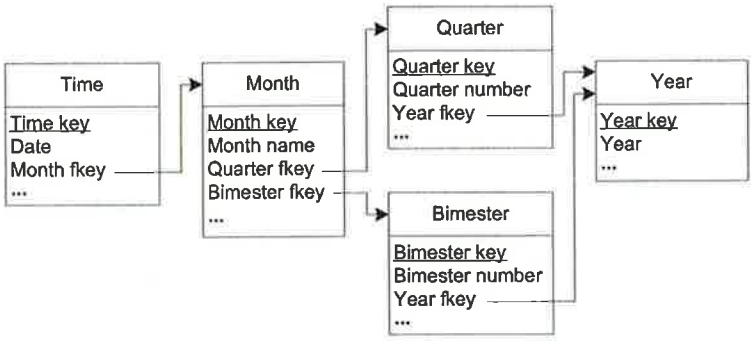


Fig. 3.40. Relations for the alternative hierarchies in Fig. 3.16

3.6.4 Parallel Hierarchies

As parallel hierarchies are composed of either alternative or simple hierarchies, their logical mapping consists in combining the mappings for the specific types of hierarchies. For example, Fig. 3.41 shows the result of applying this mapping to the relational model for the schema shown in Fig. 3.18.

Note that shared levels in parallel dependent hierarchies are represented in one table. Since this level plays different roles in each hierarchy, during implementation it is necessary to create views to avoid unnecessary repetition of data and to facilitate queries. Note also that both alternative and parallel dependent hierarchies can be easily distinguished at the conceptual level (Figs. 3.16 and 3.18); however, their logical-level representations (Figs. 3.40 and 3.41) look similar in spite of several characteristics that differentiate them, as explained in Sect. 3.2.4.

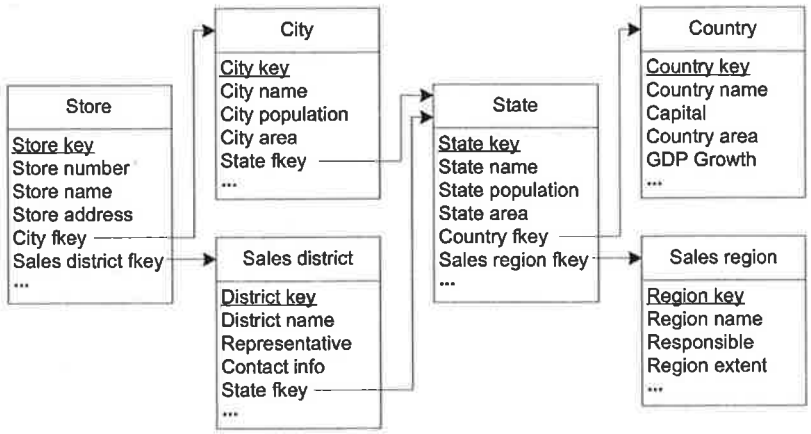


Fig. 3.41. Relational schema of a set of parallel dependent hierarchies composed of two balanced hierarchies

3.7 Implementing Hierarchies

This section describes the implementation of the various kinds of hierarchies discussed in this chapter in two commercial tools, Microsoft Analysis Services 2005 and Oracle OLAP 10g.

3.7.1 Hierarchies in Analysis Services 2005

Balanced Hierarchies

As already said in Sect. 2.10, both the snowflake and the star schema representation are supported in Analysis Services [101]. Recall that what we have called hierarchies are called in Analysis Services **user-defined hierarchies** or **multilevel hierarchies**. Multilevel hierarchies can be defined by means of all attributes in a dimension; these attributes may be stored in a single table or in several tables of a snowflake schema.

Unbalanced Hierarchies

Analysis Services proposes the implementation of unbalanced hierarchies using parent-child hierarchies. Nevertheless, this implies that the measures must be the same for all hierarchy levels.

In a parent-child hierarchy, nonleaf members may also have associated measures in a fact table, in addition to measures aggregated from their child members. For these nonleaf members, special system-generated child members are created that contain the underlying fact table data. These are referred to

as **data members**, and are always included as child members for the purposes of aggregation. However, the normal aggregation behavior can be overridden using a **custom roll-up** formula.

The MembersWithData property is used to control the visibility of data members. Data members are shown when this property is set to NonLeafDataVisible in the parent attribute; data members are hidden when this property is set to NonLeafDataHidden. The MembersWithDataCaption property of the parent attribute allows one to define a naming template, used to generate member names for data members.

Generalized Hierarchies

Analysis Services does not support generalized hierarchies. If the members differ in attributes and in hierarchy structure, the common solution is to define one hierarchy for the common levels and another hierarchy for each of the exclusive paths containing the specific levels.

On the other hand, Analysis Services supports **noncovering hierarchies**, called **ragged hierarchies**. A ragged hierarchy is defined using all its levels (i.e., the longest path). Members that do not include a particular level in the hierarchy may contain null values or empty strings, or they can contain the same value as their parent to serve as a placeholder. The dimension property HideMemberIf is used to display these members appropriately. The possible values for the HideMemberIf property and their associated behaviors are as follows:

- Never: Level members are never hidden.
- OnlyChildWithNoName: A level member is hidden when it is the only child of its parent and its name is null or an empty string.
- OnlyChildWithParentName: A level member is hidden when it is the only child of its parent and its name is the same as the name of its parent.
- NoName: A level member is hidden when its name is empty.
- ParentName: A level member is hidden when its name is identical to that of its parent.

Nonstrict Hierarchies

Analysis Services 2005 allows one to define many-to-many relationships between a dimension and a fact relationship. This is done using a bridge table such as that given in Fig. 3.38. In the case of this table, using the terminology of Analysis Services, the Section dimension has a many-to-many relationship with the Payroll measure group, through the Employee intermediate dimension and the EmplSection measure group.

Alternative Hierarchies

Several hierarchies can be attached to a dimension, and they can share levels. For example, two hierarchies can be defined on the Time dimension of Fig. 3.16: a first one composed of Time–Month–Quarter–Year, and another one composed of Time–Month–Bimester–Year.

However, there is no notion of alternative hierarchies since hierarchies cannot be defined as exclusive. For example, the two hierarchies in Fig. 3.16 can be combined, giving meaningless intersections such as Q2-2001 and B1-2001. Even though the measures are manipulated adequately, giving an empty display for these meaningless intersections, the tool should switch automatically between the exclusive component hierarchies. For example, if a user is analyzing measures using the hierarchy Time–Month–Quarter–Year and requests the hierarchy Time–Month–Bimester–Year, the first hierarchy should be excluded from the current analysis.

Parallel Hierarchies

Analysis Services supports parallel hierarchies, whether dependent or independent. Levels can be shared among the various component hierarchies.

3.7.2 Hierarchies in Oracle OLAP 10g

The general steps for defining hierarchies in Oracle OLAP 10g [214] are to create first a dimension, then the levels, and then the hierarchies, indicating the levels that compose them. In the next step, the mapping of levels to source data is performed.

As already said in Sect. 2.11, Oracle OLAP provides two kinds of hierarchies: level-based and value-based hierarchies. The former require the definition of the component levels, which usually correspond to columns in a relational table (in a flat or normalized form). On the other hand, value-based hierarchies are formed by considering relationships between members that are defined in a parent-child table.

To ensure the uniqueness of the members forming a dimension, Oracle OLAP allows designers to use natural or surrogate keys. Natural keys are read from the relational source without modification. Surrogate keys are created before the members are loaded into the analytic workspace; they consist of the prefix of the name of the level added to each member. Since levels are not defined in value-based hierarchies, Oracle OLAP imposes the restriction that only natural keys and not surrogate keys can be used with value-based hierarchies.

Balanced Hierarchies

Oracle OLAP refers to balanced hierarchies as **normal** hierarchies. They are defined as level-based hierarchies, i.e., the hierarchy structure is formed in

terms of levels and relationships between them. Oracle OLAP allows the mapping of this kind of hierarchies to both the star and the snowflake schema representation, i.e., to columns stored in a single table or in several tables.

Unbalanced Hierarchies

Unbalanced hierarchies are called **ragged hierarchies** in Oracle OLAP. Oracle OLAP provides two solutions for implementing them, depending on whether or not levels can be distinguished. One option is to define unbalanced hierarchies as level-based; this requires the inclusion of null values in the columns of the source table corresponding to "missing" child members. Another option is to define unbalanced hierarchies as value-based hierarchies, whose data comes from parent-child tables. Note that **recursive hierarchies** can be implemented only as value-based hierarchies.

Generalized Hierarchies

Currently it is not possible to manage generalized hierarchies in Oracle OLAP. They can be defined either as one hierarchy that includes the common levels, or as separate cubes, each cube having a dimension composed of a hierarchy specific to a group of members.

On the other hand, Oracle OLAP supports **noncovering hierarchies**, which are referred to as **skip-level hierarchies**. These can also be defined as either level-based or value-based hierarchies. Defining noncovering hierarchies as level-based hierarchies requires that the source table contains null values in the columns corresponding to "missing" levels. When a noncovering hierarchy is defined as value-based, a parent-child table will represent the relationships between members of the hierarchy.

Nonstrict Hierarchies

Oracle OLAP does not support nonstrict hierarchies.

Alternative Hierarchies

Oracle OLAP does not distinguish between alternative and parallel hierarchies. Therefore, even though developers can define several hierarchies within a dimension, it is the users' responsibility to know that the combination of the component hierarchies is meaningless and should not be applied.

Parallel Hierarchies

Since Oracle OLAP supports the definition of several hierarchies within the same dimension, parallel independent hierarchies can be implemented by defining each hierarchy separately and by indicating the levels composing it.

On the other hand, for parallel dependent hierarchies, each shared level must be defined separately for every hierarchy that includes it. Further, to ensure the uniqueness of members belonging to the same dimension, members of "repeated" levels should have different surrogates assigned to them.

3.8 Related Work

The advantages of conceptual modeling for database design have been acknowledged for several decades and have been studied in many publications (e.g., [30, 46]). However, the analysis presented in [262] shows the small interest of the research community in conceptual multidimensional modeling. A detailed description of multidimensional models can be found in [251, 299]. Some proposals provide graphical representations based on the ER model ([266, 302]), or on UML ([5, 301]), or propose new notations ([83, 110, 305]), while other proposals do not refer to a graphical representation ([107, 233, 245]). Other publications related to data warehouses and OLAP systems refer to the summarizability problem (e.g., [105, 106, 107, 164]), multidimensional normal forms (e.g., [161, 163]), or implementation considerations (e.g., [58, 142]), among other things.

The publications reviewed in this section fall into three categories. First, publications describing various kinds of hierarchies. Second, publications referring to logical representations of hierarchies. Finally, we briefly cover the commercial solutions for implementing hierarchies. For better comparison, in the description that follows, we use our own terminology for the various kinds of hierarchies.

There is great variation in the kinds of hierarchies supported by current multidimensional models. Some models (e.g., [83, 266, 299]) support only simple hierarchies. This situation is considered as a shortcoming of existing models for data warehouses [104]. Table 3.2 compares the multidimensional models that, to the best of our knowledge, cope with hierarchies. We have used three symbols in this table for this comparison: — when no reference to the specified kind of hierarchy exists, ± when only a description and/or definition of the hierarchy is presented, and ✓ when a description and a graphical representation are given. If a different name for a hierarchy is proposed, it is included in the table.

As can be seen in Table 3.2, the existing multidimensional models do not consider all of the kinds of hierarchies described in this book. Several models give only a description and/or a definition of some of these hierarchies, without a graphical representation (e.g., [105, 130, 208, 209, 233, 245]). Further, different names are used for the same kind of hierarchy (for example for unbalanced, generalized, and nonstrict hierarchies). Although some proposals refer to generalized hierarchies (e.g., [170]), they include the supertype and the subtypes as hierarchy levels, without considering the attributes that form a hierarchy as done in our model. All of the proposed models support strict

Table 3.2. Comparison of conceptual models that support various kinds of hierarchies

Model	Balanced	Unbalanced	Generalized	Non-covering	Nonstrict	Alternative or parallel dependent
[233]	± Onto	± Non-onto	—	± Non-covering	± Nonstrict	± Multiple
[110]	✓ Simple	—	✓ Multiple optional	—	—	✓ Alternative
[302]	✓	✓	—	—	✓ Nonstrict	✓ Multiple
[105, 107]	± Strictly homogeneous	± Homo-geneous	± Hetero-geneous	± Hetero-geneous	—	± Hetero-geneous
[245]	± Total classification	± Partial classification	± Multiple	± Multiple	—	± Multiple multiplicity
[130]	±	± Unbalanced	± Hetero-geneous	± Hetero-geneous	—	±
[16]	—	—	±	—	—	—
[5]	✓	± Non-onto	—	±	±	✓
[170]	✓	±	± Specialization	✓	✓	✓ Alternative
[305]	✓	—	—	✓	±	—
[83, 266] [299]	✓	—	—	—	—	✓
[208]	± Balanced	± Unbalanced	—	± Ragged	—	—
[209]	± Level based	± Value based	± Value based	± Value based	—	—

hierarchies explicitly or implicitly. None of the models takes into account the possibility of several different analysis criteria; in consequence, alternative and parallel hierarchies cannot be distinguished. As we have already explained in Sect. 3.2.4, this distinction is important and helps to avoid the assumption that every path in parallel hierarchies should lead to the same members of the shared level [160]. Very few models propose a graphical representation for the various hierarchies that facilitates their distinction at the schema and instance levels (e.g., [170, 266, 302]).

The Object Management Group (OMG) has proposed the Common Warehouse Model (CWM) [209] as a standard for representing data warehouse and OLAP systems. This model provides a framework for representing metadata about data sources, data targets, transformations and analysis, and processes and operations for the creation and management of warehouse data. This model is represented as a layered structure consisting of a number of sub-

models. One of these submodels, the resource layer, defines models that can be used for representing data in data warehouses and includes the relational model as one of these models. Further, the analysis layer presents a metamodel for OLAP, which includes the concepts of a dimension and a hierarchy. In the CWM, it is possible to represent all of the kinds of hierarchies specified in Table 3.2.

Current commercial OLAP tools do not allow conceptual modeling of hierarchies. They usually provide a logical-level representation limited to star or snowflake schemas. In the previous sections we have referred to publications that use the relational approach to represent the various kinds of hierarchies, where complex hierarchies are sometimes transformed into simpler ones. For example, we discussed the approaches taken by Pedersen et al. [233] for transforming unbalanced, noncovering, and nonstrict hierarchies, and the solutions proposed by Kimball and Ross [147] for managing balanced, generalized, and nonstrict hierarchies. We also discussed the advantages and disadvantages of the proposed solutions. Even though some of these solutions (e.g., [130]) provide a correct mapping that captures the semantics of the hierarchy, they produce a significant number of relations and require SQL extensions for their management. On the other hand, Song et al. [283] have proposed several logical solutions to the problem of managing many-to-many relationships between facts and dimension tables. Besides the solutions described in Sect. 3.2.2, these authors have proposed other solutions that count instances instead of aggregating the measures from a fact table.

Some commercial products, such as Microsoft Analysis Services 2005 and Oracle OLAP 10g, can cope with some of the hierarchies presented in this chapter, as we described in Sect. 3.7. On the other hand, IBM provides some other tools for implementing data warehouses and analytical applications. Alphablox Analytics [112] is used for multidimensional analysis of data represented in a relational DB2 server as a star schema, a snowflake schema, or combination of the two. Alphablox analytics allows the definition of balanced, unbalanced, and noncovering hierarchies, where the latter two hierarchies are implemented as a flat table with null values for missing members. Additionally, unbalanced hierarchies that represent recursive parent-child relationships can be stored in a parent-child table. Alphablox Analytics does not allow the definition of alternative and parallel hierarchies.

3.9 Summary

Data warehouses are defined using a multidimensional view of data, which is based on the concepts of facts, measures, dimensions, and hierarchies. OLAP systems allow users to interactively query warehouse data using operations such as drill-down and roll-up, and these operations require the definition of hierarchies for aggregating measures.

A hierarchy represents some organizational, geographic, or other type of structure that is important for analysis purposes. However, although many kinds of hierarchies can be found in real-world applications, current OLAP systems can manage only a limited number of them. This imposes important restrictions when users require complex multidimensional analysis including several kinds of hierarchies. Thus, designers must apply various implementation strategies to transform some kinds of hierarchies into simpler ones.

This chapter has presented the MultiDim model, a conceptual multidimensional model. The MultiDim model is based on the entity-relationship model and provides an intuitive graphical notation. It is well known that graphical representations facilitate the understanding of application requirements by users and designers. The model also has an associated textual notation, which is described in Appendix A. That appendix also provides a formalization of the model.

We have classified the hierarchies, taking into account their differences at the schema and at the instance level. We have distinguished between simple and alternative hierarchies. The latter are composed of one or more simple hierarchies accounting for the same analysis criterion. Moreover, simple hierarchies can be of various kinds: balanced, unbalanced, and generalized. Recursive and noncovering hierarchies are special cases of unbalanced and generalized hierarchies, respectively. For each of these simple hierarchies, another specialization can be applied, making them either strict or nonstrict. The latter allows many-to-many relationships between parent and child levels, defining graphs at the instance level. Finally, we have distinguished the situation where several hierarchies accounting for different analysis criteria may be attached to the same dimension. Depending on whether or not they share common levels, we have called these hierarchies parallel dependent and parallel independent hierarchies, respectively.

We have also provided a mapping of our conceptual model to the relational and object-relational models. For this mapping, we used well-known rules that transform entity-relationship models to logical models. We have proposed some modifications of these rules for the case of generalized and nonstrict hierarchies. We have also compared alternative relational approaches to implementing some kinds of hierarchies and given indications of when to use these alternative approaches. Finally, we have described how to implement the various kinds of hierarchies in commercial products, using as examples Microsoft Analysis Services 2005 and Oracle OLAP 10g.

The notation of the MultiDim model allows a clear distinction of each kind of hierarchy, which is not the case with most existing conceptual multidimensional models. It is important to capture this distinction at the conceptual level, since when a mapping to the logical level is performed it is no longer possible to distinguish between generalized, alternative, and parallel dependent hierarchies. Therefore, the MultiDim model allows us to capture in a better way the multidimensional semantics of data, which can then be exploited in data warehouse and OLAP systems.

Review Questions

3.1 Discuss the following concepts: dimension, level, attribute, key, fact relationship, role, measure, hierarchy, parent-child relationship, cardinalities, root level, leaf level, and analysis criterion.

3.2 Explain the difference, at the schema and at the instance level, between balanced and unbalanced hierarchies.

3.3 Give an example of a recursive hierarchy. Explain how to represent an unbalanced hierarchy with a recursive one.

3.4 Explain the usefulness of generalized hierarchies. To which concept of the entity-relationship model these hierarchies are related?

3.5 What is a splitting level? What is a joining level? Does a generalized hierarchy always have a joining level?

3.6 Explain why noncovering (or ragged) hierarchies are a particular case of generalized hierarchies.

3.7 Describe using an example how the type of a generalization in the ER model (total vs. partial, exclusive vs. overlapping) influences the corresponding generalized hierarchy in the multidimensional model.

3.8 What is the difference between strict and nonstrict hierarchies?

3.9 Illustrate with one example the problem of double counting of measures for nonstrict hierarchies. Describe different solutions to this problem.

3.10 What is a distributing factor? Explain the importance of choosing an appropriate distributing factor.

3.11 Explain in what situations alternative hierarchies are used.

3.12 Describe the difference between parallel dependent and parallel independent hierarchies.

3.13 Illustrate with examples the difference between generalized, alternative, and parallel hierarchies.

3.14 What are role-playing dimensions?

3.15 Why some dimensions are called fact or degenerate?

3.16 Relate the problem of double counting to the functional and multivalued dependencies that hold in a fact relationship.

3.17 Why a fact relationship must be decomposed in the presence of dependencies? Show an example of a fact relationship that can be decomposed differently according to the dependencies that hold on it.

3.18 Discuss the mapping rules for translating a MultiDim schema into a relational schema. Are these rules similar to those used for translating an ER schema into a relational schema?

3.19 Explain how a balanced hierarchy can be mapped into either normalized or denormalized tables. Discuss the advantages and disadvantages of these alternative mappings.

3.20 How to transform at the logical level an unbalanced hierarchy into a balanced one?

3.21 Describe different approaches for representing generalized hierarchies at the logical level.

3.22 Explain how a nonstrict hierarchy can be represented in the relational model and in the object-relational model.

3.23 Is it possible to distinguish between generalized, alternative, and parallel dependent hierarchies at the logical level?

3.24 Identify the kind of hierarchies that can be directly represented in Analysis Services and in Oracle OLAP.