Capítulo 16

Tecnología y administración de data warehouse

Objetivos de aprendizaje

Este capítulo ofrece los cimientos para una forma emergente de bases de datos, llamada data warehouse, que se usa cada vez más para apoyar la toma de decisiones. Después de este capítulo, el estudiante habrá adquirido los siguientes conocimientos y habilidades:

- Explicar las diferencias conceptuales entre las bases de datos operacionales y los data warehouse.
- Comprender arquitecturas para aplicar la tecnología de data warehouse en las organizaciones.
- Comprender la representación y manipulación de cubos de datos.
- Aplicar el modelado y la manipulación relacional de datos para datos multidimensionales.
- Explicar aspectos de la calidad de los datos y la función de las herramientas de extracción, transformación y carga para el mantenimiento de un data warehouse.
- Adquirir idea del complejo proceso de refrescar un data warehouse.

Panorama general

Imagine al ejecutivo de un corporativo nacional minorista de electrónicos. "¿Qué tiendas minoristas fueron las principales productoras durante los últimos 12 meses en la región de Rocky Mountain?". Las preguntas de seguimiento pueden incluir: "¿cuáles fueron los productos más rentables en las principales tiendas minoristas?", y "¿cuáles fueron las más exitosas promociones de productos en las principales tiendas minoristas?". Éstos son ejemplos de preguntas de apoyo a la toma de decisiones o de inteligencia de negocios que se hacen todos los días gerentes de todo el mundo. Las respuestas a estas preguntas a menudo requieren enunciados SQL que pueden tomar horas para codificarse y ejecutarse. Además, formular algunas de estas consultas puede requerir datos de un conjunto diverso de sistemas heredados internos y fuentes de mercado externas, implicando tanto bases de datos relacionales como no relacionales.

Preguntas para la toma de decisiones como las anteriores dan lugar a nuevos requerimientos de los DBMS. Este capítulo presenta la tecnología y administración de data warehouse para satisfacer los requerimientos del apoyo a la toma de decisiones. La tecnología del data warehouse

complementa y extiende la tecnología de base de datos relacionales más allá del procesamiento de transacciones en línea y las capacidades de consulta simple, como la cláusula GROUP BY en SQL. En este capítulo aprenderá inicialmente acerca de los requerimientos únicos para el procesamiento de un data warehouse, en contraposición con los requerimientos de procesamiento de transacciones que estudiamos en el capítulo 15. Después, aprenderá sobre el modelo multidimensional de datos y su implementación en bases de datos relacionales, con especial énfasis en las características de almacenamiento de datos en Oracle 10g. Aprenderá nuevas habilidades para el modelado y formulación de consultas que complementan el material de modelado de datos de la parte 3 y el material de formulación de consultas de las partes 2 y 5. Por último, aprenderá acerca del mantenimiento de un data warehouse, importante proceso que llevan a cabo los administradores de los data warehouse.

16.1 Conceptos básicos

El tipo de datos que se usa para fines de apoyo a la toma de decisiones es conceptualmente distinto al que se utiliza en las bases de datos de procesamiento de transacciones. Por lo tanto, también son diferentes las bases de datos que pueden almacenar dichos datos y las arquitecturas de cómputo que pueden procesarlos. Esta sección examina estas diferencias para proporcionar el fundamento para el estudio detallado de los aspectos tecnológicos y administrativos de secciones posteriores.

16.1.1 Procesamiento de transacciones versus apoyo a las decisiones

El procesamiento de transacciones implica diferentes necesidades en una organización que requiere apoyo a las decisiones. El procesamiento de transacciones, como se analizó en el capítulo 15, permite a las organizaciones hacer negocios diariamente de forma eficiente. Las bases de datos operacionales o de producción que se utilizan en el procesamiento de transacciones ayudan con decisiones como rastreo de pedidos, solución de quejas de los clientes y requerimientos de personal. Estas decisiones implican datos detallados acerca de los procesos empresariales.

En contraste, el procesamiento de apoyo a las decisiones ayuda a la alta gerencia a darle dirección a una organización a mediano y largo plazos. La gerencia necesita apoyo para toma de decisiones acerca de la capacidad de planeación, desarrollo de productos, localización de puntos de venta, promoción de productos y otras necesidades. Tradicionalmente, la mayoría de las organizaciones han supuesto que las bases de datos operacionales pueden proporcionar los datos para el apoyo a las decisiones. Conforme las organizaciones han desarrollado bases de datos operacionales para varias funciones, se ha desarrollado una brecha de información. Las organizaciones se han dado cuenta de modo gradual de que deben transformar significativamente las bases de datos para el apoyo a las decisiones.

Desde principios de la década de los noventa, se ha desarrollado un consenso sobre la necesidad de transformar las bases de datos operacionales para el apoyo a las decisiones. Las bases de datos operacionales pueden contener inconsistencias en áreas como formatos, identificación de entidades y unidades de medida que afectan su uso en el apoyo a las decisiones. Por otro lado, el apoyo a las decisiones necesita una visión amplia que integra procesos empresariales. Como resultado de los diferentes requerimientos, las bases de datos operacionales por lo general están separadas de las bases de datos para el apoyo a las decisiones. Utilizar una base de datos común para ambos tipos de procesamiento puede degradar en forma considerable el desempeño y hacer que sea difícil resumir la actividad en todos los procesos empresariales.

16.1.2 Características de los data warehouse

data warehouse
depósito central para datos resumidos e integrados de bases de datos operacionales y fuentes de datos externas.

El data warehouse, término creado por William Inmon en 1990, se refiere a un depósito de datos central donde se integran, depuran y estandarizan los datos de bases de datos operacionales y otras fuentes para apoyar la toma de decisiones. Las actividades de transformación (depuración, integración y estandarización) son esenciales para lograr beneficios. Los beneficios tangibles de un data warehouse pueden incluir mayores ingresos y menores gastos permitidos por el análisis

empresarial, que no eran posibles antes de la utilización de los data warehouse. Por ejemplo, un data warehouse puede permitir menores pérdidas gracias a la mejor detección de fraudes, mejor retención de clientes a través del marketing dirigido y reducción en los costos implícitos de inventario por medio de un mejor pronóstico de la demanda.

Los requerimientos de procesamiento de las aplicaciones de apoyo a las decisiones han llevado a cuatro características distintivas para los data warehouse, mismas que son descritas a continuación:

1. **Orientado a los sujetos:** Un data warehouse se organiza en torno de los principales sujetos o entidades, como clientes, pedidos y productos. Esta orientación a los sujetos contrasta con la mayor orientación hacia los procesos en el procesamiento de transacciones.
2. **Integrado:** Los datos operativos de múltiples bases de datos y fuentes de datos externas se integran en un data warehouse para proporcionar una base de datos individual y unificada para el apoyo a las decisiones. La consolidación de los datos requiere convenciones de nomenclatura consistente, formatos de datos uniformes y escalas de medición comparables en las bases de datos y en las fuentes de datos externas.
3. **Variante de tiempo:** Los data warehouse usan marcas de tiempo para representar datos históricos. La dimensión del tiempo es crítica para identificar tendencias, pronosticar operaciones futuras y establecer objetivos de operación. Los data warehouse esencialmente consisten en una extensa serie de fotografías instantáneas, cada una de las cuales representa datos operativos capturados en un momento en el tiempo.
4. **No volátil:** Los datos nuevos en un data warehouse se anexan, en vez de reemplazarse, de modo que se preservan los datos históricos. El acto de anexar datos nuevos se conoce como refrescar el data warehouse. La falta de operaciones de actualización y eliminación garantiza que un data warehouse no contiene anomalías de actualización o eliminación. Los datos de las transacciones se transfieren a un data warehouse sólo cuando se ha completado la mayoría de las actividades de actualización.

La tabla 16.1 ilustra mejor las características de los data warehouse en comparación con las bases de datos operacionales. El procesamiento de transacciones depende de bases de datos operacionales con datos actuales en el plano individual, en tanto que el procesamiento de apoyo a las decisiones utiliza data warehouse con datos históricos tanto en el plano individual como en el resumido. Los datos del plano individual proporcionan flexibilidad para responder a una amplia variedad de necesidades de apoyo a las decisiones, mientras que los datos resumidos dan respuesta rápida a consultas repetitivas. Por ejemplo, una transacción de captura de pedidos requiere datos acerca de clientes individuales, pedidos y artículos en el inventario, en tanto que una aplicación de apoyo a las decisiones puede usar ventas mensuales a los clientes durante un periodo de varios años. Por consiguiente, las bases de datos operacionales tienen una orientación hacia los procesos (por ejemplo, todos los datos relevantes para un proceso empresarial particular), en comparación con una orientación hacia los sujetos de los data warehouse (por ejemplo, todos los datos de los clientes o todos los datos de los pedidos). Una transacción generalmente actualiza sólo unos cuantos registros, mientras que una aplicación de apoyo a las decisiones puede hacer consultas entre miles o millones de registros.

TABLA 16.1
Comparación de bases de datos operacionales y los data warehouse

Característica	Base de datos operacional	Data warehouse
Actualidad	Actual	Histórico
Nivel de detalle	Individual	Individual y resumen
Orientación	Orientación hacia los procesos	Orientación hacia los sujetos
Número de registros procesados	Unos cuantos	Miles
Nivel de normalización	Normalizada en su mayoría	Violaciones frecuentes de la BCNF
Nivel de actualización	Volátil	No volátil (refrescado)
Modelo de datos	Relacional	Modelo relacional con esquemas de estrella y modelo multidimensional con cubos de datos

La integridad de los datos y el desempeño del procesamiento de transacciones requiere que las bases de datos operacionales estén muy normalizadas. En contraste, los data warehouse por lo regular son desnormalizados por la Forma Normal Boyce-Codd para reducir el esfuerzo de unir tablas grandes. La mayor parte del procesamiento de un data warehouse implica recuperaciones e inserciones periódicas de nuevos datos. Estas operaciones no sufren de anomalías causadas por un diseño que no esté totalmente normalizado.

Debido a los diferentes requerimientos de procesamiento, se han desarrollado distintos modelos de datos para las bases de datos operacionales y los data warehouse. El modelo de datos relacional predomina en el caso de las bases de datos operacionales. Durante los primeros años de uso del data warehouse dominaba el modelo multidimensional de datos. En años recientes, las bases de datos relacionales se han empleado para los data warehouse con un patrón de esquema conocido como esquema de estrella. El modelo de datos multidimensional se utiliza ahora como una representación para el usuario final de la vista de un data warehouse.

16.1.3 Arquitecturas para data warehouse

A pesar de los beneficios potenciales de un data warehouse, muchos proyectos de data warehouse han fracasado como resultado de una planeación deficiente. Los proyectos de data warehouse son grandes esfuerzos que implican coordinación entre muchas partes de una organización. Muchas organizaciones han subestimado el tiempo y el esfuerzo para conciliar diferentes vistas de un data warehouse. Además, el verdadero tamaño de un data warehouse puede llevarlo a un desempeño deficiente. Una arquitectura apropiada puede ayudar a aliviar problemas con el desempeño del data warehouse y el esfuerzo de desarrollo.

modelo de datos empresariales (EDM)
modelo de datos conceptual del data warehouse que define la estructura del data warehouse y los metadatos para tener acceso y transformar las bases de datos operacionales y las fuentes de datos externas.

Para la mayoría de las organizaciones, es apropiada una arquitectura de data warehouse de dos o tres niveles. En una arquitectura de data warehouse de dos niveles (figura 16.1), los datos operativos se transforman y luego se transfieren a un data warehouse. Puede emplearse un plano separado de servidores para soportar las complejas actividades del proceso de transformación. Para asistir con el proceso de transformación, se crea un modelo de datos empresariales (EDM). El EDM describe la estructura del data warehouse y contiene los metadatos requeridos para entrar a bases de datos operacionales y fuentes de datos externas. El EDM también puede contener detalles acerca de la depuración e integración de las fuentes de datos. La gerencia usa el data warehouse directamente para recuperar datos para el apoyo a las decisiones.

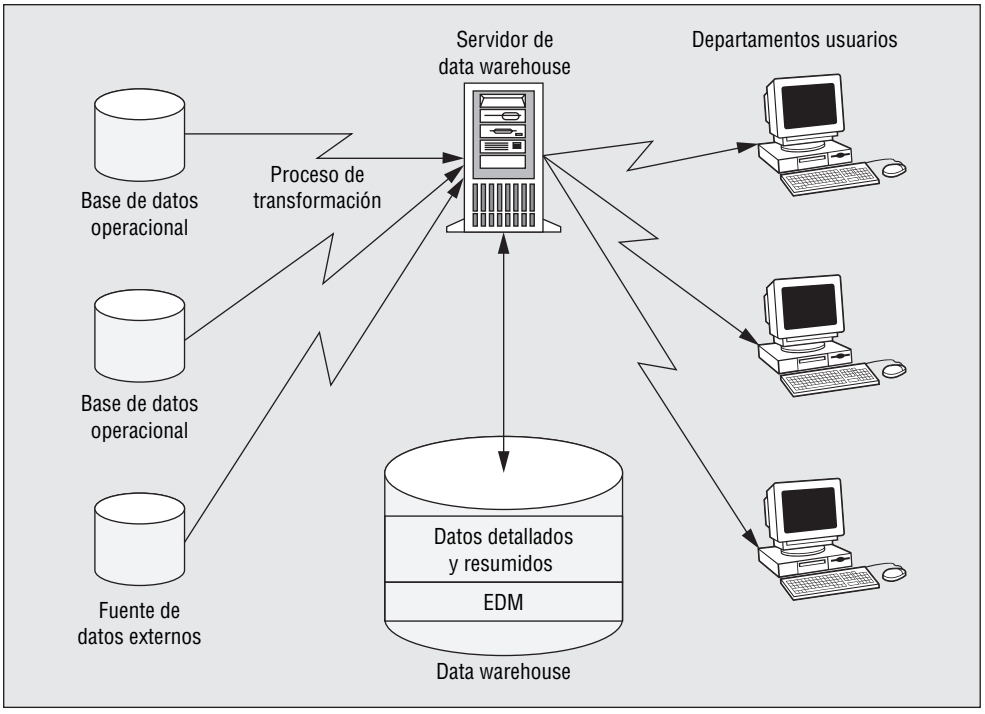


FIGURA 16.1
Arquitectura de data warehouse de dos niveles

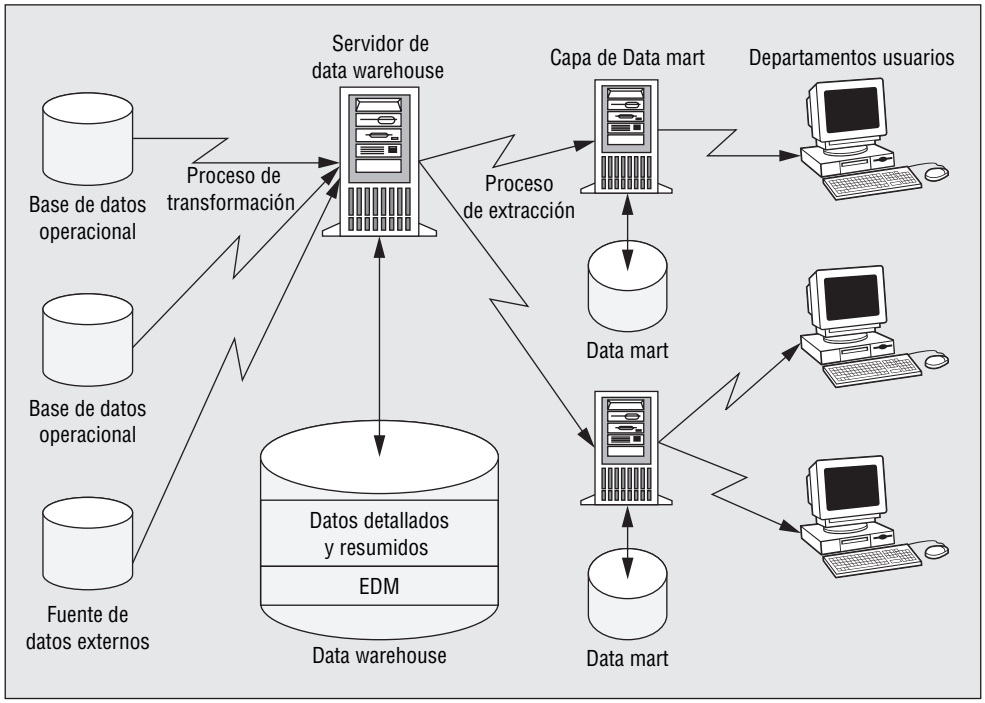


FIGURA 16.2
Arquitectura de data warehouse de tres niveles

La arquitectura de dos niveles puede tener problemas de desempeño para data warehouse grandes con aplicaciones intensivas de datos para el apoyo a las decisiones. Para resolver estas dificultades, muchas organizaciones grandes utilizan una arquitectura de tres niveles de data warehouse, como se muestra en la figura 16.2. Los usuarios departamentales generalmente necesitan acceso a pequeñas porciones del data warehouse y no al almacén entero. Para proporcionarles rápido acceso mientras se les aísla de los datos que necesitan otros grupos de usuarios, con frecuencia se usan data warehouse más pequeños llamados data marts. Los data marts actúan como la interfaz entre los usuarios finales y el data warehouse corporativo, almacenando un subconjunto del data warehouse y refrescándolo periódicamente (por ejemplo, diaria o semanalmente). Por lo regular, el data warehouse y los data marts residen en diferentes servidores para mejorar el desempeño y la tolerancia a errores. Los usuarios departamentales retienen el control de sus propios data marts, en tanto que el data warehouse permanece bajo control del personal de sistemas de información de la corporación.

data mart
subconjunto o vista de un data warehouse, normalmente en un departamento o nivel funcional, que contiene todos los datos requeridos para las tareas del apoyo a las decisiones de ese departamento.

Dado el contexto de generalización en toda la empresa de un data warehouse, algunas organizaciones creen que la construcción de un data mart puede ser un retraso innecesario con oportunidades de negocios perdidas si se pone demasiado énfasis en crear primero un modelo de datos empresariales. En su lugar, algunas organizaciones emplean un planteamiento ascendente, como se ilustra en la figura 16.3. En una arquitectura ascendente de data warehouse, los datos se modelan una entidad a la vez y se les almacena en data marts separados. Con el paso del tiempo, se sintetizan, depuran y fusionan nuevos datos en data marts existentes o se les integra en nuevos data marts. El conjunto de data marts puede evolucionar a un data warehouse grande si la organización puede justificar la erogación de construir un modelo de datos empresariales.

Un desarrollo más reciente es el surgimiento de oper marts (abreviación de mercados operacionales –operational mart–), como lo señala Imhoff (2001). Un mercado operacional es un data mart justo-a-tiempo, normalmente construido a partir de una base de datos operacional en anticipación o en respuesta a eventos mayores, como desastres e introducciones de nuevos productos. Un mercado operacional da soporte a la demanda pico para los reportes y análisis empresarial que acompañan a un evento mayor. Después de que se atiende la demanda de apoyo a las decisiones, se puede desmantelar el mercado operativo o se puede fusionar en un data warehouse existente.

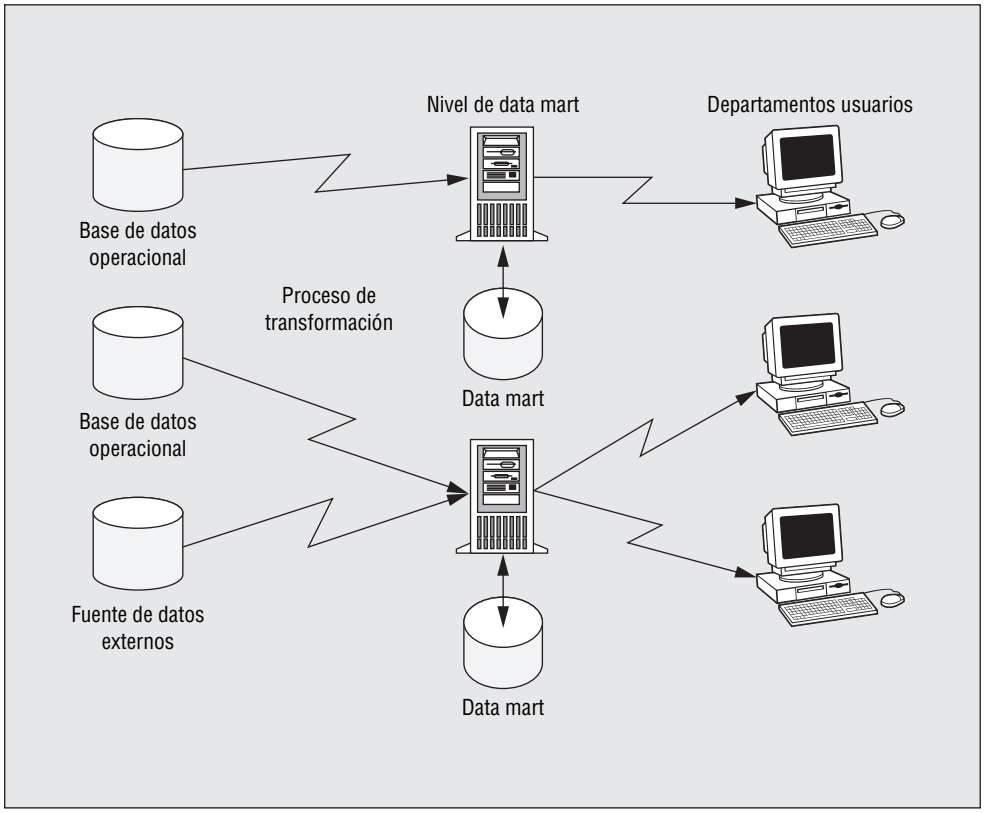


FIGURA 16.3
Arquitectura ascendente de data warehouse

16.1.4 Minería de datos

minería de datos
proceso de descubrimiento de patrones implícitos en datos almacenados en un data warehouse y el uso de dichos patrones para una ventaja empresarial.

Los data warehouse mejoran la calidad de la toma de decisiones al consolidar y agregar datos de transacciones. Es posible aumentar el valor de un data warehouse si se pueden descubrir patrones ocultos en los datos. La minería de datos se refiere al proceso de descubrimiento de patrones implícitos en datos almacenados en un data warehouse y la utilización de esos patrones como una ventaja empresarial. La minería de datos facilita la capacidad de detectar, entender y pronosticar patrones.

La aplicación más común de las técnicas de minería de datos es el marketing de objetivo. Las compañías de ventas por correo pueden incrementar los ingresos y disminuir los costos si pueden identificar a probables clientes y eliminar a los clientes con pocas probabilidades de compra. Las técnicas de minería de datos permiten a las personas que toman decisiones enfocarse en los esfuerzos de marketing por los datos demográficos y psicográficos de los clientes. Las industrias minorista, bancaria, de viajes y de bienes de consumo también han aprovechado las técnicas de minería de datos. Por ejemplo, la industria minorista usa técnicas de minería de datos para dirigir promociones y cambiar la combinación y orden de los artículos en las tiendas.

La minería de datos está considerada como un anexo de un data warehouse maduro. La minería de datos necesita datos más detallados que los que proporcionan los data warehouse tradicionales. Los volúmenes de datos y su dimensionalidad pueden ser mucho mayores en el caso de las técnicas de minería de datos que en los de las otras herramientas de análisis del data warehouse. Las técnicas de minería de datos se enriquecen con datos depurados de transacciones altamente dimensionales. Para soportar estos requerimientos de minería de datos, muchos data warehouse ahora almacenan datos detallados en el nivel del cliente individual, producto y otros.

La minería de datos requiere una serie de herramientas que se extienden más allá de las herramientas de análisis estadístico tradicional. Las herramientas de análisis estadístico tradicional no son adecuadas para datos muy dimensionales con una mezcla de datos numéricos y categóricos. Además, las técnicas estadísticas tradicionales no escalan bien a grandes cantidades de datos. La minería de datos incluye generalmente los siguientes tipos de herramientas:

- Herramientas de acceso a datos para extraer y muestrear datos de transacciones de acuerdo con criterios complejos para grandes bases de datos.
- Herramientas de visualización de datos que permiten que una persona que toma decisiones adquiera una comprensión más profunda e intuitiva de los datos.
- Una rica colección de modelos para agrupar, pronosticar y determinar reglas de asociación de grandes cantidades de datos. Los modelos implican redes neuronales, algoritmos genéticos, inducción en árboles de decisiones, algoritmos para descubrir reglas, redes de probabilidad y otras tecnologías de sistema experto.
- Una arquitectura que proporciona optimización, procesamiento cliente-servidor y consultas paralelas para escalar a grandes cantidades de datos.

Como complemento para el almacenamiento de datos, la minería de datos ofrece ideas que pueden eludir las técnicas tradicionales. La minería de datos sostiene la promesa de influir de manera más eficaz en los data warehouse al dar la capacidad para identificar relaciones ocultas en los datos que ahí se almacenan; facilita un descubrimiento guiado por los datos, usando técnicas como la creación de reglas de asociación (por ejemplo, entre el presupuesto para publicidad y las ventas de temporada), generación de perfiles (por ejemplo, patrones de compra para un segmento de clientes específico) y demás. Se puede emplear este conocimiento para mejorar las operaciones empresariales en áreas críticas, habilitar esfuerzos de marketing de objetivo y mejorar el servicio al cliente así como para la detección de fraudes.

16.1.5 Aplicaciones de los data warehouse

Los proyectos de data warehouse se emprenden generalmente por razones competitivas: con el fin de lograr una ventaja estratégica o seguir siendo competitivos. Es frecuente que se emprenda un proyecto de data warehouse como parte de una estrategia corporativa para cambiar de un enfoque en el producto a un enfoque en el cliente. Los data warehouse exitosos han ayudado a identificar nuevos mercados, concentrar los recursos en clientes rentables, mejorar la retención de clientes y reducir los costos de inventario. Después del éxito de las organizaciones precursoras, otras organizaciones han seguido el ejemplo para permanecer en niveles competitivos.

Los proyectos de data warehouse se han emprendido en una gran variedad de industrias. Unas cuantas aplicaciones clave han llevado a la adopción de proyectos de data warehouse, como se enlista en la tabla 16.2. Las industrias altamente competitivas, como la venta al detalle, seguros, aerolíneas y telecomunicaciones (particularmente de servicio de larga distancia), han invertido con anticipación en tecnología y proyectos de data warehouse. Las industrias menos competitivas, como las de servicios públicos reguladas, han tardado más en invertir, a pesar de que incrementan las inversiones conforme maduran la tecnología y la práctica del data warehouse.

La madurez en la utilización de data warehouse varía entre industrias y organizaciones. Los primeros en adoptar data warehouse los han desplegado desde principios de la década de los noventa, mientras que quienes los adoptaron después lo hicieron hasta finales de la misma década. Con el rápido desarrollo de la tecnología de data warehouse y mejores prácticas, se necesita de una inversión continua en estos rubros con el objeto de sostener el valor de la empresa. Para ofrecer una guía acerca de las decisiones de inversión y mejores prácticas, las organizaciones se interesan en comparaciones con organizaciones semejantes para medir el nivel de uso del data warehouse.

TABLA 16.2
Aplicaciones del almacenamiento de datos por industria

Industria	Aplicaciones clave
Aerolíneas	Administración de rendimiento, evaluación de rutas
Telecomunicaciones	Retención de clientes, diseño de redes
Seguros	Evaluación de riesgos, diseño de productos, detección de fraudes
Venta al detalle	Marketing de objetivo, administración de la cadena de abastecimiento

TABLA 16.3
Etapas del modelo de maduración del data warehouse

Fuente: Eckerson 2004.

Etapa	Alcance	Arquitectura	Uso administrativo
Prenatal	Sistema operativo	Informes de administración	Centro de costos
Infantil	Analistas empresariales individuales	Hojas de cálculo	Discernimientos de administración
Niñez	Departamentos	Mercados de datos	Apoyo al análisis empresarial
Adolescencia	Divisiones	Data warehouse	Rastro de procesos empresariales
Edad adulta	Empresa	Data warehouse empresarial	Guía de la organización
Sabiduría	Inter–empresas	Servicios web y redes externas	Guía del mercado y la industria

El modelo de madurez del data warehouse se ha propuesto con el fin de ofrecer una guía para las decisiones de inversión en data warehouse (Eckerson 2004). El modelo de madurez consiste de seis etapas, las cuales se resumen en la tabla 16.3. Las etapas proporcionan un marco para observar el progreso de una organización, no una métrica absoluta, ya que las organizaciones pueden demostrar aspectos de múltiples etapas al mismo tiempo. Conforme las organizaciones pasan de etapas inferiores a más avanzadas, puede incrementarse el valor de la empresa. No obstante, a las organizaciones se les puede dificultar la justificación de nuevas inversiones importantes en data warehouse en las etapas de adolescencia y edad adulta, puesto que en ocasiones no es fácil cuantificar los beneficios.

Un aspecto importante del modelo de madurez es la dificultad de moverse entre ciertas etapas. En el caso de las organizaciones pequeñas pero en crecimiento, pasar de la infancia a la etapa de niñez puede ser difícil porque se necesita una inversión significativa en tecnología de data warehouse. Por lo que respecta a las grandes organizaciones, se lucha por pasar de la adolescencia a la edad adulta. Para hacer la transición, la alta gerencia debe percibir el data warehouse como un recurso empresarial vital, no sólo como una herramienta que proporciona el departamento de tecnología de la información.

16.2 Representación multidimensional de los datos

Después de entender los requerimientos únicos de datos para el apoyo a las decisiones, usted está listo para aprender acerca de la tecnología para satisfacer los requerimientos. El modelo multidimensional de datos soporta la representación y operación de datos especialmente diseñados para el procesamiento de apoyo a las decisiones en los data warehouse. Esta sección describe la terminología y operaciones del modelo multidimensional de datos.

16.2.1 Ejemplo de un cubo de datos multidimensional

Considere una compañía que vende productos electrónicos en distintas partes de Estados Unidos. En parte, la compañía comercializa cuatro diferentes productos de impresión (láser monocromática, inyección de tinta, para fotografías y portátil) en cinco estados diferentes (California, Washington, Colorado, Utah y Arizona). Para almacenar los datos de las ventas diarias para cada producto y cada ubicación en la base de datos relacional, necesita la tabla 16.4, la cual consta de tres columnas (*Product*, *Location* y *Sales)* y 20 filas (cuatro casos de *Product* por cinco casos de *Location*).

La representación de la tabla 16.4 puede ser compleja y pesada. Primero, imagine que la compañía desea agregar un quinto producto (digamos, láser a color). Con el fin de rastrear las ventas por estados de este nuevo producto, necesita agregar cinco filas, una por cada estado. En segundo lugar, debemos hacer notar que los datos de la tabla 16.4 representan datos de las ventas para un día en particular (por ejemplo, 10 de agosto de 2006). Para almacenar los mismos datos para el total de los 365 días del año 2006, necesita añadir una cuarta columna para almacenar la fecha de las ventas y duplicar las 20 filas para cada fecha 365 veces, con el objeto de

TABLA 16.4
Representación relacional de los datos de ventas

Product	Location	Sales
Mono Laser	California	80
Mono Laser	Utah	40
Mono Laser	Arizona	70
Mono Laser	Washington	75
Mono Laser	Colorado	65
Ink Jet	California	110
Ink Jet	Utah	90
Ink Jet	Arizona	55
Ink Jet	Washington	85
Ink Jet	Colorado	45
Photo	California	60
Photo	Utah	50
Photo	Arizona	60
Photo	Washington	45
Photo	Colorado	85
Portable	California	25
Portable	Utah	30
Portable	Arizona	35
Portable	Washington	45
Portable	Colorado	60

TABLA 16.5
Representación multidimensional de los datos de ventas

	Product			
Location	Mono Laser	Ink Jet	Photo	Portable
California	80	110	60	25
Utah	40	90	50	30
Arizona	70	55	60	35
Washington	75	85	45	45
Colorado	65	45	85	60

sumar una cuarta columna, para dar un total de 7 300 filas. De igual modo, si quiere almacenar datos históricos para un periodo de 10 años, requiere 73 000 filas. Cada nueva fila debe contener los valores de producto, estado y fecha.

Un análisis de los datos de la tabla 16.4 revela que los datos contienen dos dimensiones, *Product* y *Location,* y un valor numérico para las ventas unitarias. Por lo tanto, es posible simplificar conceptualmente la tabla 16.4 al reordenar los datos en un formato multidimensional (véase tabla 16.5).

Es sencillo comprender y extender la representación multidimensional. Por ejemplo, sumar un quinto producto requiere una columna adicional a la derecha de la tabla 16.4. Para añadir fechas, es necesaria un tercera dimensión llamada *Time*, dando como resultado un arreglo tridimensional, como se muestra en la figura 16.4. Puede conceptualizar esta tabla tridimensional como un libro que consta de 365 páginas, cada página almacena datos de las ventas por producto y estado para una fecha específica del año. Además, la tabla multidimensional es más compacta porque las etiquetas de fila y columna no se duplican como en la tabla 16.3.

La representación multidimensional también ofrece una conveniente muestra de los totales sumatorios. Cada dimensión en un cubo puede contener totales que un usuario puede identificar con facilidad (totales de fila, totales de columna, totales de profundidad y totales generales). Por ejemplo, para añadir totales de fila a la tabla 16.5, puede agregarse una columna *Totals* con un valor por fila como se muestra en la tabla 16.6. En la representación relacional que se ilustra en la tabla 16.4, es preciso añadir totales utilizando valores nulos para los valores de columna. Por ejemplo, para representar el total de ventas de California para todos los productos, tiene que sumarse la fila <−, California, 275> a la tabla 16.4, donde el signo − indica todos los productos.

FIGURA 16.4
Cubo de datos tridimensional

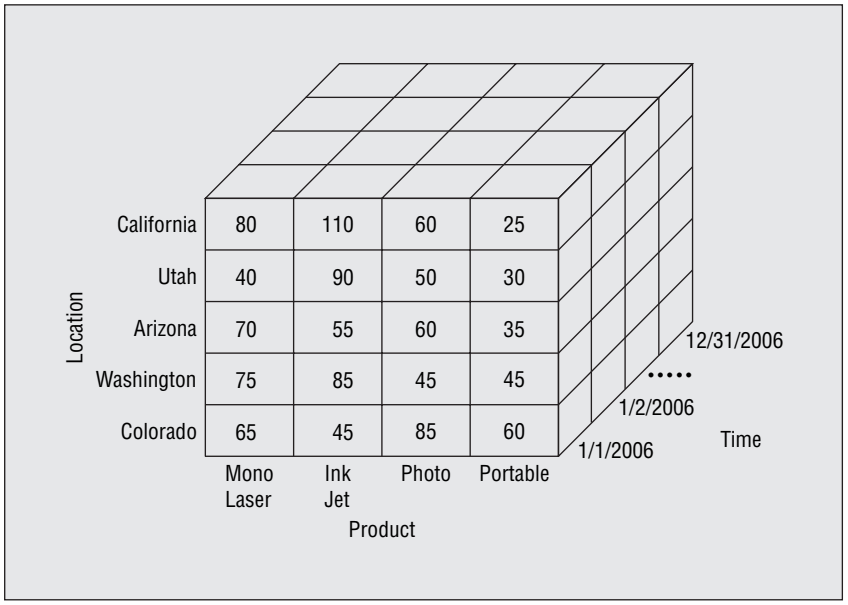


TABLA 16.6
Representación multidimensional de los datos de ventas con totales de fila

	Product				
Location	**Mono Laser**	**Ink Jet**	**Photo**	**Portable**	**Totals**
California	80	110	60	25	275
Utah	40	90	50	30	210
Arizona	70	55	60	35	220
Washington	75	85	45	45	250
Colorado	65	45	85	60	255

Además de las ventajas de utilidad, la representación multidimensional puede proporcionar una mayor velocidad de recuperación. El almacenamiento directo de datos multidimensionales pone en evidencia la necesidad de convertir una representación de tabla en una representación multidimensional. No obstante, la representación multidimensional puede sufrir almacenamiento excesivo porque muchas celdas pueden permanecer vacías. Aun con técnicas de compresión, las grandes tablas multidimensionales pueden consumir considerablemente más espacio de almacenamiento que las tablas relacionales correspondientes.

En síntesis, una representación multidimensional proporciona una interfaz intuitiva para los analistas empresariales. Conforme aumenta el número de dimensiones, los analistas empresariales encuentran una representación multidimensional fácil de entender y visualizar en comparación con una representación relacional. Como la representación multidimensional satisface las necesidades de los analistas empresariales, ésta se emplea en gran medida en las herramientas de reportes empresariales, aun cuando las tablas relacionales ofrecen almacenamiento histórico.

cubo de datos
formato multidimensional en el que las celdas contienen datos numéricos, llamados medidas, organizados por temas, denominados dimensiones. En ocasiones, al cubo de datos se le conoce como hipercubo porque conceptualmente puede tener un número ilimitado de dimensiones.

16.2.2 Terminología multidimensional

Un cubo de datos generaliza las representaciones bidimensional (véase tabla 16.5) y tridimensional (véase figura 16.4) que estudiamos en la sección anterior. Un cubo de datos consiste en celdas que contienen medidas (valores numéricos, como las cantidades de ventas unitarias) y dimensiones para etiquetar o agrupar datos numéricos (por ejemplo, *Product, Location* y *Time*). Cada dimensión contiene valores que reciben el nombre de miembros. Por ejemplo, la dimensión *Location* tiene cinco miembros en la tabla 16.4 (California, Washington, Utah, Arizona y

Colorado). Tanto las dimensiones como las medidas pueden almacenarse o derivarse. Por ejemplo, la fecha de compra es una dimensión almacenada con el año, el mes y el día de las compras como dimensión derivada.

Detalles de dimensión

Las dimensiones pueden tener jerarquías compuestas por niveles. Por ejemplo, la dimensión *Location* puede tener una jerarquía compuesta por los niveles país, estado y ciudad. De igual forma, la dimensión *Time* puede tener una jerarquía compuesta por año, trimestre, mes y fecha. Pueden usarse jerarquías para descender de los niveles superiores al detalle (por ejemplo, país) y trabajar en dirección opuesta. A pesar de que las jerarquías no son esenciales, permiten una representación conveniente y eficiente. Sin jerarquías, la dimensión *Location* debe contener el nivel más detallado (ciudad). Sin embargo, esta representación puede ser difícil para calcular totales en toda la dimensión. Alternativamente, es posible dividir la dimensión *Location* en dimensiones separadas para país, estado y ciudad, resultando en un cubo más grande.

Con fines de flexibilidad, las dimensiones pueden tener múltiples jerarquías. En una dimensión con múltiples jerarquías, por lo regular se comparte por lo menos un nivel. Por ejemplo, la dimensión *Location* puede tener una jerarquía con los niveles país, estado y ciudad y una segunda jerarquía con los niveles país, estado y código postal. La dimensión *Time* puede tener una jerarquía con los niveles año, trimestre y fecha, y una segunda jerarquía con los niveles año, semana y fecha. Las jerarquías múltiples permiten organizaciones alternativas para una dimensión.

Otra característica de la dimensión es la jerarquía irregular para una relación autorreferenciada entre miembros del mismo nivel. Por ejemplo, una dimensión de gerente podría tener una jerarquía irregular para mostrar las relaciones entre miembros y subordinados. Un analista que manipula un cubo de datos tal vez quiera expandir o contraer la dimensión de gerente de acuerdo con las relaciones entre gerentes y subordinados.

La selección de dimensiones influye en la dispersión de un cubo de datos. La dispersión indica el grado de celdas vacías en un cubo de datos. La dispersión puede ser un problema si se relacionan dos o más dimensiones. Por ejemplo, puede haber celdas vacías si ciertos productos se venden sólo en estados seleccionados. Si existe una gran cantidad de celdas vacías, el cubo de datos puede desperdiciar espacio y hacer que el proceso sea más lento. Es posible utilizar técnicas de compresión especial para reducir el diseño de los cubos de datos dispersos.

Detalles de medida

Las celdas de un cubo de datos contienen medidas como los valores de ventas de la figura 16.4. Las medidas soportan operaciones numéricas como aritmética simple, cálculos estadísticos y ecuaciones simultáneas. Una celda puede contener una o más medidas. Por ejemplo, el número de unidades puede ser otra medida para el cubo de datos de ventas. El número de celdas ocupadas en un cubo multidimensional debe equivaler al número de filas en la tabla relacional correspondiente (la tabla 16.5 contiene 20 celdas ocupadas que corresponden a 20 filas de la tabla 16.4).

Las medidas derivadas pueden almacenarse en un cubo de datos o calcularse a partir de otras medidas en tiempo de operación. Las medidas que se pueden derivar de otras medidas en la misma celda normalmente no se almacenarían. Por ejemplo, el total de ventas en dólares puede calcularse como el total de ventas unitarias por las medidas de precio unitario en una celda. Las medidas de resumen derivadas de un serie de celdas pueden almacenarse o calcularse dependiendo del número de celdas y del costo de tener acceso a las celdas para el cálculo.

Otros ejemplos de cubo de datos

Como indica esta sección, los cubos de datos pueden extenderse más allá del ejemplo de tres dimensiones que se muestra en la figura 16.4. La tabla 16.7 lista cubos de datos comunes para soportar la administración de recursos humanos y el análisis financiero. Las dimensiones con diagonales indican dimensiones jerárquicas. Las dimensiones de tiempo y ubicación también son jerárquicas, pero los posibles niveles no se listan porque pueden ser específicos para la organización.

TABLA 16.7
Cubos de datos para soportar la administración de recursos humanos y el análisis financiero

Cubo de datos	Dimensiones comunes	Medidas comunes
Análisis del cambio de personal	Compañía/línea de negocio/ departamento, ubicación, rango salarial, clasificación del puesto, tiempo	Búsqueda de personal para contrataciones, transferencias, terminaciones y retiros
Utilización de empleados	Compañía/línea de negocio/ departamento, ubicación, rango salarial, clasificación del puesto, tiempo	Horas completas equivalentes (FTE; *full time equivalent*), horas FTE normales, horas FTE de tiempo extra
Análisis de activos	Tipo de activo, años en banda de servicio, tiempo, cuenta, compañía/línea de negocio/departamento, ubicación	Costo, valor neto en libros, valor mercantil
Análisis de fabricantes	Fabricantes, ubicación, cuenta, tiempo, unidad empresarial	Importe total de la facturación

16.2.3 Datos de serie de tiempo

El tiempo es una de las dimensiones más comunes en un data warehouse y es útil para capturar tendenciasy hacer pronósticos. Una serie de tiempo proporciona almacenamiento de todos los datos históricos en una celda, en vez de especificar una dimensión de tiempo separada. La estructura de una medida se hace más compleja con una serie de tiempo, pero el número de dimensiones se reduce. Además, muchas funciones estadísticas pueden operar directamente en los datos de la serie de tiempo.

Una serie de tiempo es un tipo de datos de arreglo con varias propiedades especiales, como se mencionan a continuación. El arreglo soporta una serie de valores, uno para cada periodo. Los ejemplos de series de tiempo incluyen cantidades de ventas semanales, precios diarios de cierres de acciones y salarios anuales de los empleados. La siguiente lista muestra las propiedades típicas de una serie de tiempo:

- **Tipo de datos:** Esta propiedad denota la clase de dato almacenado en los puntos de datos. Normalmente, el tipo de datos es numérico, como números de punto flotante, números decimales fijos o enteros.
- **Fecha de inicio:** Esta propiedad expresa la fecha de inicio del primer punto de datos; por ejemplo, 1/1/2006.
- **Calendario:** Esta propiedad contiene el año calendario apropiado para la serie de tiempo; por ejemplo, el año fiscal 2006. Un conocimiento extensivo de las reglas del calendario, como determinar años bisiestos y días festivos integrados en el calendario, reduce el esfuerzo en el desarrollo del data warehouse.
- **Periodicidad:** Esta propiedad especifica el intervalo entre los puntos de datos. La periodicidad puede ser diaria, semanal, mensual, trimestral, anual (calendario o años fiscales), por hora, intervalos de 15 minutos, periodicidad personalizada, entre otros.
- **Conversión:** Esta propiedad especifica la conversión de datos unitarios en datos agregados. Por ejemplo, agregar ventas diarias en las ventas semanales requiere total agregado, mientras que añadir precios de acciones diarios en los precios semanales implica una operación de promedio.

16.2.4 Operaciones del cubo de datos

Para los cubos de datos se han propuesto varias operaciones de apoyo a las decisiones. Esta sección analiza las operaciones que se emplean con mayor frecuencia. Aún está en desarrollo un conjunto estándar de operaciones de cubo de datos y no todas las herramientas de data warehouse soportan actualmente todas las operaciones.

Rebanada

El cubo de datos puede contener un gran número de dimensiones y los usuarios a menudo necesitan concentrarse en un subconjunto de dimensiones para lograr algún discernimiento. El

FIGURA 16.5
Ejemplo de operación de rebanada

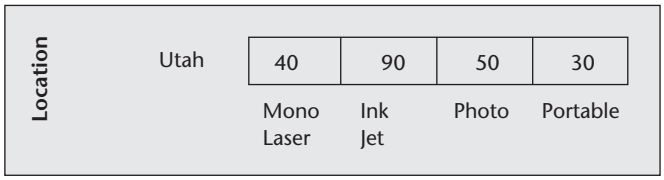
(Ubicación X rebanada de producto para la fecha = 1/1/2006)

	Product			
Location	**Mono Laser**	**Ink Jet**	**Photo**	**Portable**
California	80	110	60	25
Utah	40	90	50	30
Arizona	70	55	60	35
Washington	75	85	45	45
Colorado	65	45	85	60

FIGURA 16.6
Ejemplo de operación de resumen de rebanada

	Time			
Location	**1/1/2006**	**1/2/2006**	. . .	**Total Sales**
California	400	670	. . .	16,250
Utah	340	190	. . .	11,107
Arizona	270	255	. . .	21,500
Washington	175	285	. . .	20,900
Colorado	165	245	. . .	21,336

FIGURA 16.7
Ejemplo de operación de dado



operador de rebanada recupera un subconjunto de datos similar al operador de restricción del álgebra relacional. En una operación de rebanada, se establecen una o más dimensiones en valores específicos y se despliega el cubo de datos restante. Por ejemplo, la figura 16.5 muestra el cubo de datos resultante de la operación de rebanada en el cubo de datos de la figura 16.4, donde Time = 1/1/2006 junto con otras dos dimensiones (*Location* y *Product*).

Una variación del operador de rebanada permite que una persona que toma decisiones resuma a todos los miembros en vez de enfocarse sólo en uno. El operador de resumen de rebanada reemplaza una o más dimensiones con cálculos de resumen. El cálculo de resumen con frecuencia indica el valor total de todos los miembros o la tendencia central de la dimensión como promedio o valor mediano. Por ejemplo, la figura 16.6 muestra el resultado de la operación de resumen de rebanada donde la dimensión *Product* se sustituye con la suma de las ventas de todos los productos. Puede agregarse una nueva columna llamada *Total Sales* para almacenar ventas generales de los productos para el año entero.

Dado

Como las dimensiones individuales pueden contener un gran número de miembros, los usuarios necesitan enfocarse en un subconjunto de ellos para lograr algún discernimiento. El operador de dado reemplaza una dimensión con un subconjunto de valores de la dimensión. Por ejemplo, la figura 16.6 presenta el resultado de una operación de dado para desplegar las ventas del Estado de Utah para el 1 de enero de 2006. Las operaciones de dado por lo regular se dan después de una operación de rebanada y regresa un subconjunto de los valores desplegados en la rebanada previa. Ayudan a concentrar la atención en una o más columnas de número de una rebanada.

Descenso

Los usuarios con frecuencia quieren navegar entre los niveles de las dimensiones jerárquicas. El operador de descenso permite a los usuarios navegar de un nivel más general a un nivel más

específico. Por ejemplo, la figura 16.8 muestra una operación de descenso en el Estado de Utah de la dimensión *Location*. El signo más en Utah indica una operación de descenso.

Ascenso

El ascenso es lo contrario del descenso. El ascenso implica moverse de un nivel específico a un nivel más general de una dimensión jerárquica. Por ejemplo, una persona que toma decisiones puede ascender datos de las ventas de un nivel diario a uno trimestral para las necesidades de reportes de fin de trimestre. En el ejemplo de las ventas de impresoras, la figura 16.5 presenta un ascenso del Estado de Utah de la figura 16.8.

Pivote

El operador de pivote soporta la reorganización de las dimensiones del cubo de datos. Por ejemplo, en la figura 16.8, la posición de las dimensiones *Product* y *Location* se puede revertir de modo que *Product* aparezca en las filas y *Location* en las columnas. El operador de pivote hace posible que se presente un cubo de datos con un orden visual más atractivo.

El operador de pivote se usa con mayor frecuencia en cubos de datos con más de dos dimensiones, en los que aparecen múltiples dimensiones en el área de la fila y/o columna porque no es posible desplegar dos dimensiones de otra manera. Por ejemplo, para desplegar un cubo de datos con las dimensiones *Location, Product* y *Time,* puede desplegarse la dimensión *Time* en el área de la fila dentro de la dimensión *Location.* Una operación de pivote podría reordenar el cubo de datos de forma tal que la dimensión *Location* se despliegue dentro de la dimensión *Time*.

Resumen de operadores

Para ayudarle a recordar las operaciones del cubo de datos, la tabla 16.8 resume el propósito de cada operador.

FIGURA 16.8
Operación de descenso para el Estado de Utah en la figura 16.5

	Product			
Location	**Mono Laser**	**Ink Jet**	**Photo**	**Portable**
California	80	110	60	25
+ Utah				
Salt Lake	20	20	10	15
Park City	5	30	10	5
Ogden	15	40	30	10
Arizona	70	55	60	35
Washington	75	85	45	45
Colorado	65	45	85	60

TABLA 16.8 **Resumen de los operadores del cubo de datos**

Operador	Propósito	Descripción
Rebanada	Concentrar la atención en un subconjunto de dimensiones	Reemplazar una dimensión con un valor de miembro individual o con un resumen de sus valores de medida
Dado	Enfocar la atención en un subconjunto de valores del miembro	Sustituir una dimensión con un subconjunto de miembros
Descenso	Obtener más detalles acerca de una dimensión	Navegar de un nivel más general a un nivel más específico de una dimensión jerárquica
Ascenso	Resumir detalles sobre una dimensión	Navegar de un nivel más específico a un nivel más general de una dimensión jerárquica
Pivote	Permitir que se presente un cubo de datos en un orden visualmente atractivo	Reordenar las dimensiones de un cubo de datos

16.3 Soporte de DBMS relacional para data warehouse

El modelo multidimensional de los datos descrito en la sección anterior se implementó originalmente con el propósito particular de almacenamiento para cubos de datos. Estos mecanismos de almacenamiento multidimensional soportan la definición, manipulación y optimización de grandes cubos de datos. Como consecuencia del dominio comercial de la tecnología de base de datos relacional, sólo era cuestión de tiempo antes de que los DBMSs relacionales ofrecieran soporte para los datos multidimensionales. En años recientes, los principales fabricantes de DBMS han hecho fuertes inversiones en investigación y desarrollo para soportar los datos multidimensionales. Gracias al nivel de inversión y al poder de mercado de los fabricantes de DBMS relacional, la mayoría de los data warehouse ahora usan DBMSs relacionales, por lo menos en parte.

Esta sección presenta características de los DBMSs relacionales para soportar los datos multidimensionales. Las características incluyen planteamientos de modelado de datos, representación de dimensión, extensiones para la cláusula GROUP BY, vistas materializadas con reescritura de consultas y estructuras de almacenamiento y técnicas de optimización especializadas. Para proporcionar un contexto específico para las características, algunos ejemplos usan Oracle 10g SQL.

16.3.1 Modelado de datos relacionales para datos multidimensionales

esquema de estrella
representación de modelado de datos para bases de datos multidimensionales. En una base de datos relacional, un esquema de estrella tiene una tabla de hechos en el centro relacionada con múltiples tablas de dimensión en relaciones 1-M.

Al utilizar una base de datos relacional para un data warehouse, se necesita una nueva técnica de modelado de datos a fin de representarlos de manera multidimensional. Un esquema de estrella es una representación del modelado de datos de cubos de datos multidimensionales. En una base de datos relacional, un diagrama de esquema de estrella se ve como una estrella con una tabla central grande en su centro, denominada tabla de hechos, que se vincula con múltiples tablas de dimensión de manera radial. La tabla de hechos almacena datos numéricos (hechos), como los resultados de las ventas, en tanto que las tablas de dimensión almacenan datos descriptivos correspondientes a dimensiones individuales del cubo de datos, como producto, ubicación y tiempo. Hay una relación 1-M de cada tabla de dimensión con la tabla de hechos.

La figura 16.9 muestra un esquema de estrella ERD para el ejemplo de ventas que se presentó en la sección 16.2. Este ERD consiste en cuatro tipos de entidad de dimensión: *Item*, *Customer, Store* y *TimeDim,* junto con un tipo de entidad de hecho llamado *Sales*. Al convertirse a un diseño de tabla, la tabla *Sales* tiene llaves foráneas para cada tabla de dimensión (*Item, Customer, Store* y *TimeDim*). El tipo de entidad *Item* proporciona datos para la dimensión *Product* que se muestra en los ejemplos de la sección 16.2, mientras que el tipo de entidad *Store* proporciona datos para la dimensión *Location.*

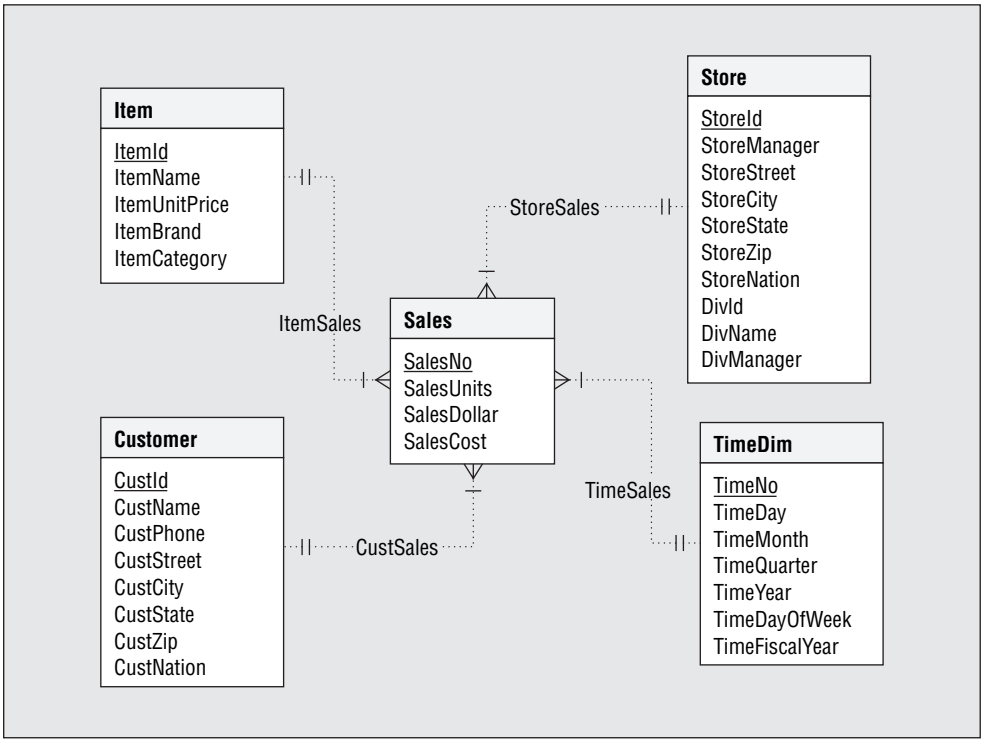
El ERD de ventas en tienda de la figura 16.9, ofrece detalles finos para un data warehouse. El ERD de ventas de tienda ofrece detalles del cliente, tienda y artículo individual. Este nivel de detalle no es necesario para soportar los cubos de datos de ventas que se presentan en la sección 16.2. Sin embargo, un nivel de detalle de profundidad ofrece flexibilidad para soportar análisis no anticipados, así como aplicaciones de minería de datos. Este nivel de detalle profundo puede copiar datos en las bases de datos operacionales, aunque la representación del data warehouse puede diferir sustancialmente, dada la orientación al sujeto del data warehouse y la depuración e integración realizadas en los datos de origen.

esquema de constelación
representación del modelado de datos para bases de datos multidimensionales. En una base de datos relacional, un esquema de constelación contiene múltiples tablas de hechos en el centro relacionadas con tablas de dimensión. Normalmente, las tablas de hechos comparten algunas tablas de dimensión.

Variaciones al esquema de estrella

El esquema de estrella de la figura 16.9 representa sólo un proceso empresarial individual para el rastreo de las ventas. Puede requerirse esquemas de estrella adicionales sólo para otros procesos, como envíos y compras. Para procesos empresariales relacionados que comparten algunas tablas de dimensión, un esquema de estrella puede extenderse en un esquema de constelación con tipos de entidad de factor múltiple, como se muestra en la figura 16.10. Al convertirse en un diseño de tabla, el tipo de entidad *Inventory* se convierte en una tabla de hechos y las relaciones 1-M se convierten en llaves foráneas en la tabla de hechos. El tipo de entidad *Inventory* agrega

FIGURA 16.9
Esquema de estrella ERD para el ejemplo de ventas de la tienda



varias medidas, incluida la cantidad disponible de un artículo, el costo del artículo y la cantidad devuelta. Todas las tablas de dimensión se comparten entre ambas tablas de hechos, excepto para las tablas *Supplier* y *Customer.*

Las tablas de hechos por lo general son normalizadas, mientras que las tablas de dimensión a menudo no están en la tercera forma normal. Por ejemplo, el tipo de entidad *Store* en las figuras 16.9 y 16.10 no está en 3NF porque *DivId* determina *DivName* y *DivManager*. Por lo regular, no es necesario normalizar las tablas de dimensión para evitar anomalías de almacenamiento porque generalmente son estables y pequeñas. La naturaleza de un data warehouse indica que deberían diseñarse tablas de dimensión para la recuperación, no para la actualización. El desempeño de la recuperación se mejora al eliminar las operaciones de enlace que se necesitarían para combinar tablas de dimensión totalmente normalizadas.

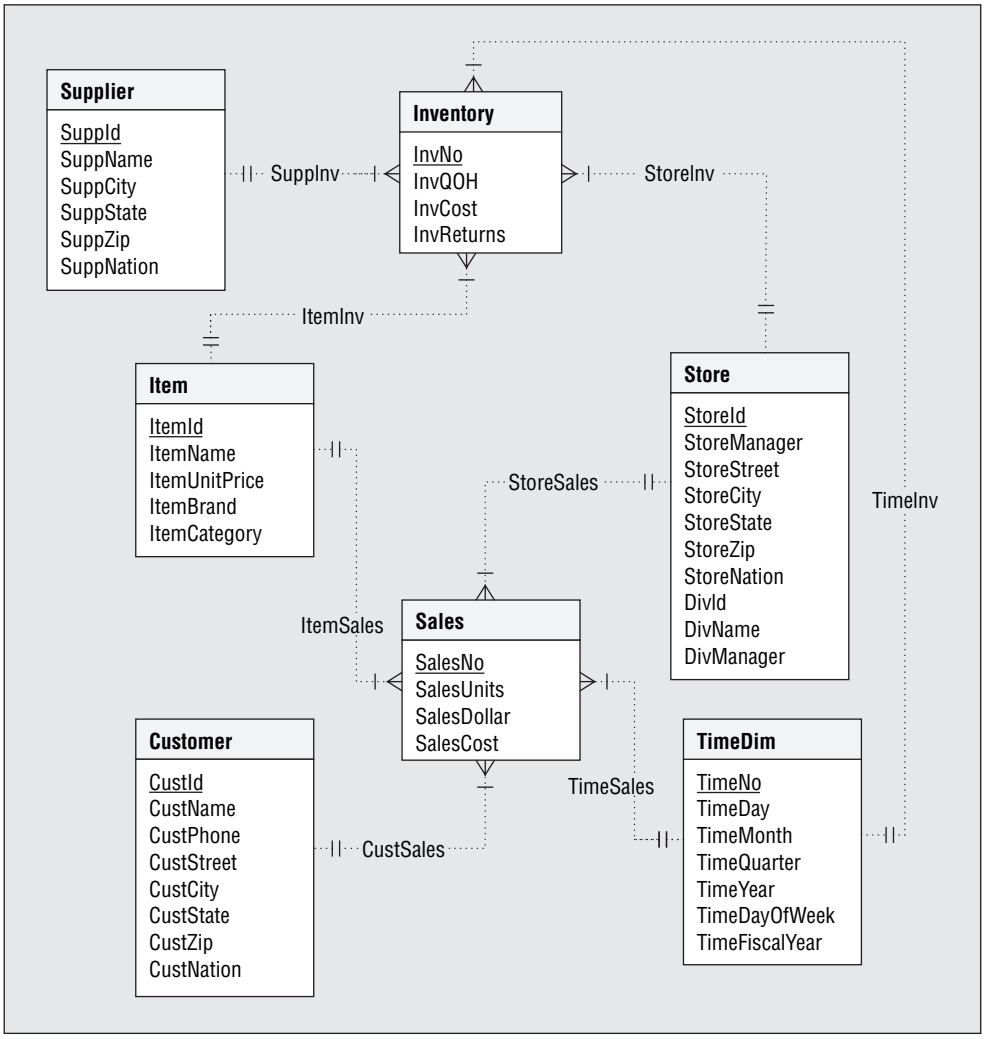
Cuando las tablas de dimensión son pequeñas, la desnormalización sólo ofrece una pequeña ganancia en el desempeño de la recuperación. Por consiguiente, es común ver tablas en pequeña dimensión normalizadas, como se aprecia en la figura 16.11. Esta variación se conoce como esquema de copo de nieve porque múltiples niveles de tablas de dimensión rodean la tabla de hechos. En el caso de las tablas *Customer* e *Item,* la normalización completa quizá no sea una buena idea porque estas tablas pueden contener muchas filas.

esquema de copo de nieve
representación del modelado de datos para bases de datos multidimensionales. En una base de datos relacional, un esquema de copo de nieve tiene niveles múltiples de tablas de dimensión en relación con una o más tablas de hechos. Debe considerar el uso del esquema de copo de nieve en lugar del esquema de estrella para tablas de poca dimensión que no estén en 3NF.

El esquema de estrella y sus variaciones requieren relaciones 1-M entre las tablas de dimensión y la tabla de hechos. El uso de relaciones 1-M simplifica la formulación de consultas y soporta las técnicas de optimización que estudiamos en la sección 16.3.5. En ocasiones, los datos de origen tienen excepciones que implican relaciones M-N, no 1-M. Por ejemplo, si la tabla de hechos *Sales* se deriva de las facturas de clientes, algunas facturas pueden implicar múltiples clientes, como compañeros de cuarto o cónyuges. En seguida se explican dos maneras de revisar el esquema de estrella para relaciones M-N.

- Si hay una cantidad pequeña y fija de clientes posibles, puede hacerse un ajuste simple en la tabla de hechos o de dimensiones. Es posible agregar múltiples columnas a la tabla de hechos o de dimensiones para permitir más de un cliente. Por ejemplo, la tabla *Customer* puede tener una columna adicional *CustId2* para identificar a un segundo cliente opcional en la factura.

FIGURA 16.10
Esquema de constelación ERD para el ejemplo de inventario de ventas



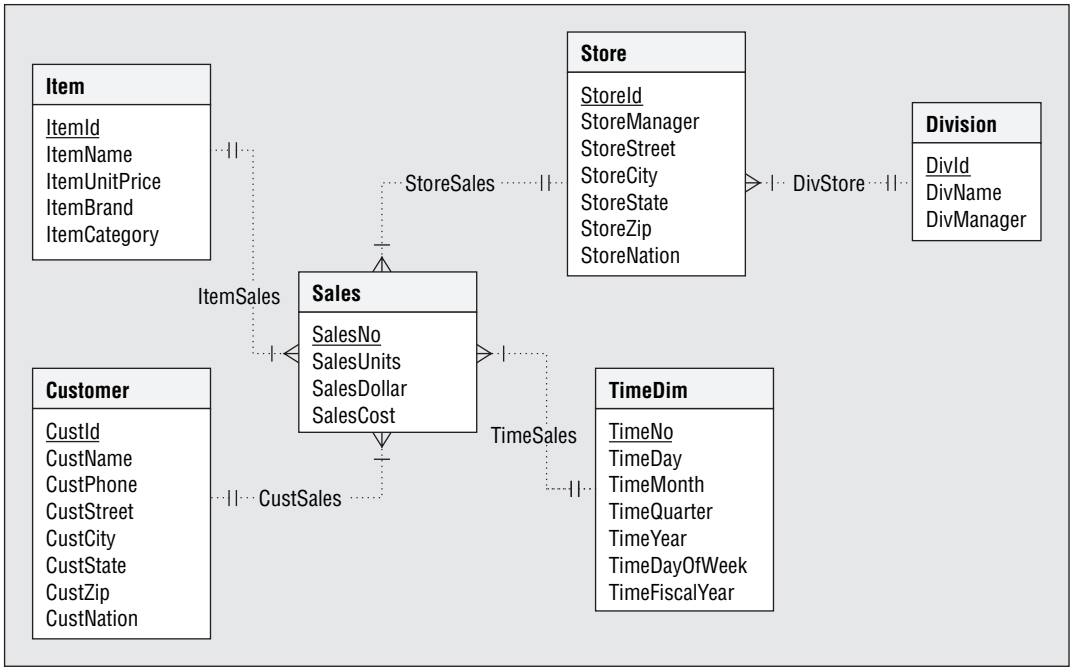
- La representación es más difícil para el caso en que haya grupos de clientes para una factura. Puede añadirse una tabla de grupo de clientes con una tabla asociativa que conecte al grupo de clientes y la tabla *Customer* a través de relaciones 1-M.

Representación temporal en los esquemas de estrella

La representación temporal es un aspecto crucial para los data warehouse porque la mayoría de las consultas usan el tiempo en las condiciones. El principal uso del tiempo es el registro de la cantidad de hechos. La representación más sencilla es un tipo de datos de estampa de tiempo para una columna en una tabla de hechos. Muchos data warehouse utilizan una llave foránea en una tabla de dimensión de tiempo en lugar de una columna de estampa de tiempo, como se muestra en las figuras 16.9 a 16.11. El uso de una tabla de dimensión de tiempo soporta una conveniente representación de las características de calendario específicas para la organización, como vacaciones, años fiscales y número de semanas que no están representadas en los tipos de datos de estampa de tiempo. La granularidad de la tabla de dimensión de tiempo por lo general se presenta en días. Si también se requiere la hora del día para una tabla de hechos, puede agregarse como una columna en la tabla de hechos a fin de aumentar la llave foránea para la tabla de tiempo.

La mayor parte de las tablas de hechos implican el tiempo representado como una llave foránea para la tabla de tiempo con el aumento para la fecha del día si es necesaria. Por lo que respecta a las tablas de hechos que implican operaciones internacionales, es posible utilizar dos

FIGURA 16.11 **Esquema de copo de nieve ERD para el ejemplo de ventas de tienda**



representaciones temporales (llaves foráneas de la tabla de tiempo junto con columnas opcionales para la hora del día) para registrar la hora en los sitios de origen y destino. Una variación identificada por Kimball (2003) es la tabla de hechos acumulativos que registra el estado de múltiples eventos en vez de un solo evento. Por ejemplo, una tabla de hechos que contiene una instantánea del procesamiento de pedidos podría incluir fecha del pedido, fecha de envío, fecha de entrega, fecha de pago y otras. Cada columna de cantidad de eventos puede representarse por medio de una llave foránea para la tabla de tiempo, junto con una columna de hora del día en caso de ser necesario.

Para las tablas de dimensión, la representación temporal implica el nivel de integridad histórica, un aspecto para las actualizaciones de las tablas de dimensión. Cuando se actualiza una fila de dimensión, las filas de la tabla de hechos relacionadas ya no son históricamente precisas. Por ejemplo, si la columna ciudad de una fila de clientes cambia, las filas de ventas relacionadas ya no son históricamente precisas. Con el fin de preservar la integridad histórica, las filas de ventas relacionadas deben apuntar hacia una versión más antigua de la fila de clientes. Kimball (abril de 1996) presenta tres alternativas para la integridad histórica:

- **Tipo I:** Sobrescribe valores antiguos con los datos modificados. Este método no ofrece integridad histórica.
- **Tipo II:** Usa un número de versión para aumentar la llave primaria de una tabla de dimensiones. Para cada cambio en una fila de dimensiones, inserte una fila en la tabla de dimensiones con un número de versión más alto. Por ejemplo, para manejar el cambio en la columna de ciudad, hay una fila nueva en la tabla *Customer* con el mismo número de cliente pero un número de versión mayor que la fila anterior. Además del número de versión, se necesitan columnas adicionales para registrar la fecha de inicio efectiva y la fecha de terminación efectiva para cada columna histórica.
- **Tipo III:** Usa columnas adicionales para mantener un historial fijo. Por ejemplo, con objeto de mantener un historial de la ciudad actual y los dos cambios previos de ciudad, pueden almacenarse tres columnas de ciudad en la tabla *Customer* (*CustCityCurr*, *CustCityPrev, CustCityPast*), junto con seis columnas de fecha asociadas (dos columnas de fecha por columna de valor histórico) para registrar las fechas efectivas.

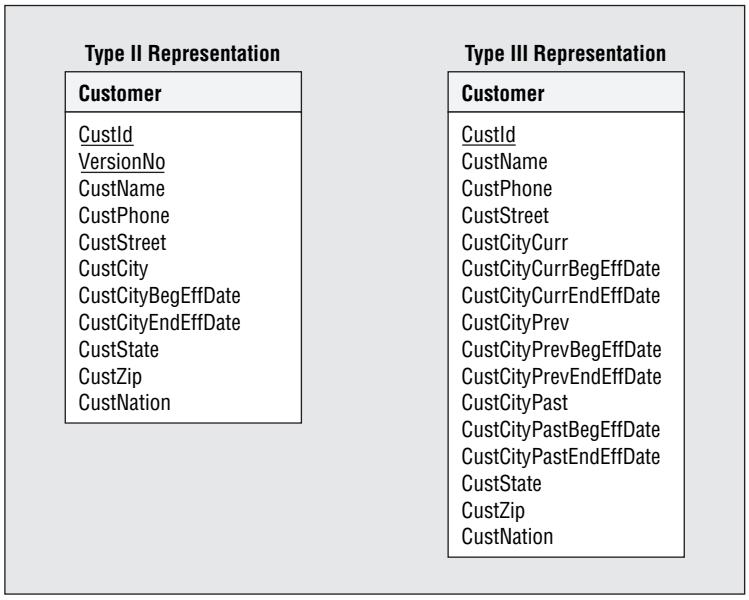


FIGURA 16.12
Alternativas para la integridad dimensional histórica de *CustCity*

La figura 16.12 muestra las alternativas tipo II y tipo III para la columna *CustCity.* La alternativa tipo II implica múltiples filas para el mismo cliente, pero se representa el historial entero.

La alternativa tipo III implica sólo una fila individual para cada cliente, pero únicamente puede representarse un historial limitado.

16.3.2 Representación de dimensión

El esquema de estrella y sus variaciones no proporcionan una representación explícita de las dimensiones jerárquicas porque las tablas de tiempo no definen las relaciones jerárquicas entre los niveles de una dimensión. Dado que la definición de dimensión es importante para soportar las operaciones del cubo de datos, así como las técnicas de optimización para la reescritura de consultas (véase sección 16.3.4), muchos fabricantes de DBMS relacional han creado extensiones de SQL propietarias para las dimensiones. Esta sección revisa el enunciado de Oracle CREATE DIMENSION para indicar los tipos de extensiones que se pueden encontrar en los DBMSs relacionales.

El enunciado de Oracle CREATE DIMENSION soporta la especificación de niveles, jerarquías y restricciones para una dimensión.[1] La primera parte de una declaración de dimensión implica la especificación de niveles. Para dimensiones planas (no jerárquicas), solamente hay un nivel individual en una dimensión. No obstante, muchas dimensiones implican niveles múltiples,

EJEMPLO 16.1 **Enunciado CREATE DIMENSION de Oracle para la dimensión *StoreDim* con la especificación de niveles**

```
CREATE DIMENSION StoreDim
  LEVEL StoreId      IS Store.StoreId
  LEVEL City         IS Store.StoreCity
  LEVEL State        IS Store.StoreState
  LEVEL Zip          IS Store.StoreZip
  LEVEL Nation       IS Store.StoreNation ;
```

[1] No ponga líneas en blanco en los enunciados CREATE DIMENSION. El compilador SQL de Oracle genera mensajes de error cuando encuentra líneas en blanco en estos enunciados.

como se ilustra en el ejemplo 16.1 para la dimensión *StoreDim*. Cada nivel corresponde a una columna de la tabla de origen *Store.*

La siguiente parte de un enunciado CREATE DIMENSION implica la especificación de jerarquías. El enunciado CREATE DIMENSION de Oracle soporta dimensiones con múltiples jerarquías, como se muestra en el ejemplo 16.2. La especificación de una jerarquía se dirige del nivel más detallado al nivel más general. Las palabras clave CHILD OF indican las relaciones de jerarquía directa en una dimensión.

EJEMPLO 16.2 **Enunciado CREATE DIMENSION de Oracle para la dimensión *StoreDim* con la especificación de niveles y jerarquías**

```
CREATE DIMENSION StoreDim
  LEVEL StoreId          IS Store.StoreId
  LEVEL City             IS Store.StoreCity
  LEVEL State            IS Store.StoreState
  LEVEL Zip              IS Store.StoreZip
  LEVEL Nation           IS Store.StoreNation
  HIERARCHY CityRollup   (
    StoreId CHILD OF
    City    CHILD OF
    State   CHILD OF
    Nation  )
HIERARCHY ZipRollup    (
    StoreId CHILD OF
    Zip     CHILD OF
    State   CHILD OF
    Nation  );
```

El enunciado CREATE DIMENSION de Oracle soporta dimensiones con niveles a partir de múltiples tablas de origen. Esta característica se aplica a las tablas de dimensiones normalizadas en esquemas de copo de nieve. El ejemplo 16.3 aumenta el ejemplo 16.2 con la inclusión de un nivel adicional (*DivId*) junto con una jerarquía adicional que contiene el nuevo nivel. En la especificación de nivel, el nivel *DivId* hace referencia a la tabla *Division.* En la jerarquía *DivisionRollup*, la cláusula JOIN KEY indica un enlace entre las tablas *Store* y *Division.* Cuando una jerarquía tiene niveles de más de una tabla de origen, se requiere la cláusula JOIN KEY al final de la especificación de jerarquías.

EJEMPLO 16.3 **Enunciado CREATE DIMENSION de Oracle para la dimensión *StoreDim* con el uso de múltiples tablas de origen**

```
CREATE DIMENSION StoreDim
  LEVEL StoreId          IS Store.StoreId
  LEVEL City             IS Store.StoreCity
  LEVEL State            IS Store.StoreState
  LEVEL Zip              IS Store.StoreZip
  LEVEL Nation           IS Store.StoreNation
  LEVEL DivId            IS Division.DivId
```

```
  HIERARCHY CityRollup    (
    StoreId CHILD OF
    City      CHILD OF
    State    CHILD OF
    Nation    )
HIERARCHY ZipRollup    (
    StoreId CHILD OF
    Zip       CHILD OF
    State    CHILD OF
    Nation    )
HIERARCHY DivisionRollup    (
    StoreId CHILD OF
    DivId
    JOIN KEY Store.DivId REFERENCES DivId    );
```

La parte final de un enunciado CREATE DIMENSION implica la especificación de restricciones. La cláusula ATTRIBUTE define relaciones de dependencia funcional que implican niveles de dimensión y columnas de no origen en tablas de dimensiones. El ejemplo 16.4 muestra cláusulas ATTRIBUTE para las columnas de no origen en la tabla *Division.*

EJEMPLO 16.4 **Enunciado CREATE DIMENSION de Oracle para la dimensión *StoreDim* con el uso de cláusulas ATTRIBUTE para las restricciones**

```
CREATE DIMENSION StoreDim
  LEVEL StoreId          IS Store.StoreId
  LEVEL City             IS Store.StoreCity
  LEVEL State            IS Store.StoreState
  LEVEL Zip              IS Store.StoreZip
  LEVEL Nation           IS Store.StoreNation
  LEVEL DivId            IS Division.DivId
  HIERARCHY CityRollup    (
    StoreId CHILD OF
    City      CHILD OF
    State    CHILD OF
    Nation    )
HIERARCHY ZipRollup    (
    StoreId CHILD OF
    Zip       CHILD OF
    State    CHILD OF
    Nation    )
HIERARCHY DivisionRollup    (
    StoreId CHILD OF
    DivId
   JOIN KEY Store.DivId REFERENCES DivId    )
ATTRIBUTE DivId DETERMINES Division.DivName
ATTRIBUTE DivId DETERMINES Division.DivManager ;
```

En el ejemplo 16.4, las cláusulas DETERMINES son redundantes con la restricción de llave primaria para la tabla *Division.* Las cláusulas DETERMINES se muestran en el ejemplo 16.4 para reforzar las restricciones soportadas por las declaraciones de llaves primarias. Las

cláusulas DETERMINES se requieren para restricciones que no corresponden con las restricciones de llaves primarias para permitir la optimización de las consultas. Por ejemplo, si cada código postal se asocia con un estado, debería utilizarse una cláusula DETERMINES para permitir optimizaciones que implican columnas de código postal y estado.

16.3.3 Extensiones para la cláusula GROUP BY para datos multidimensionales

Hay disponibles nuevas capacidades de resumen en la cláusula GROUP BY, comenzando en SQL: 1999 y continuando con SQL: 2003. Estas características son un intento por unificar la proliferación de extensiones propietarias de cubo de datos, aunque no ponen en evidencia la necesidad de herramientas visuales para soportar directamente las operaciones del cubo de datos. Las extensiones implican la capacidad de producir totales sumatorios (operadores CUBE y ROLLUP), así como la especificación más precisa de las columnas de agrupación (operadores GROUPING SETS). Esta sección describe estas nuevas partes de la cláusula GROUP BY usando Oracle 10g como ejemplo de DBMS que implementa las características estándar de SQL.

Operador CUBE

operador CUBE operador que aumenta el resultado normal GROUP BY con todas las combinaciones de subtotales. El operador CUBE es apropiado para resumir columnas de dimensiones independientes más que columnas que representen diferentes niveles de una sola dimensión.

La cláusula del operador CUBE produce todas las combinaciones de subtotales además de los totales normales que se muestran en una cláusula GROUP BY. Debido a que se generan todos los subtotales posibles, el operador CUBE es apropiado para resumir columnas de dimensiones independientes más que columnas que representen diferentes niveles de la misma dimensión. Por ejemplo, el operador CUBE podría ser apropiado para generar subtotales para todas las combinaciones de mes, estado de la tienda y marca del artículo. En contraste, una operación CUBE tendría interés limitado para mostrar todos los subtotales posibles de año, mes y día, como resultado de la jerarquía en la dimensión de tiempo.

Para ilustrar el operador CUBE, el ejemplo 16.5 presenta el enunciado SELECT con una cláusula GROUP BY que contiene sólo dos columnas. En el resultado aparecen sólo seis filas, de modo que puede comprenderse con facilidad el efecto del operador CUBE. Con dos valores en la columna *StoreZip* y tres valores en la columna *TimeMonth*, el número de combinaciones

EJEMPLO 16.5 **Cláusula GROUP BY y resultado parcial sin subtotales**

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId AND
            Sales.TimeNo = TimeDim.TimeNo
      AND (StoreNation = 'USA' OR StoreNation = 'Canada')
      AND TimeYear = 2005
GROUP BY StoreZip, TimeMonth;
```

StoreZip	TimeMonth	SumSales
80111	1	10000
80111	2	12000
80111	3	11000
80112	1	9000
80112	2	11000
80112	3	15000

EJEMPLO 16.6 (Oracle) **Cláusula GROUP BY y resultado con subtotales producidos por el operador CUBE**

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
      AND Sales.TimeNo = TimeDim.TimeNo
      AND (StoreNation = 'USA' OR StoreNation = 'Canada')
      AND TimeYear = 2005
  GROUP BY CUBE(StoreZip, TimeMonth);
```

StoreZip	TimeMonth	SumSales
80111	1	10000
80111	2	12000
80111	3	11000
80112	1	9000
80112	2	11000
80112	3	15000
80111		33000
80112		35000
	1	19000
	2	23000
	3	26000
		68000

del subtotal es seis (dos subtotales *StoreZip,* tres subtotales *TimeMonth* y un gran total), como se aprecia en el ejemplo 16.6. Los valores en blanco en el resultado representan un resumen sobre todos los valores posibles de la columna. Por ejemplo, la fila <80111, −, 33000> representa las ventas totales en el código postal 80111 en todos los meses (− representa un valor sin interés para el mes).

Con más de dos columnas de agrupación, se hace más difícil entender el operador CUBE. Los ejemplos 16.7 y 16.8 extienden los ejemplos 16.5 y 16.6 con una columna de agrupación adicional (*TimeYear*). El número de filas en el resultado aumenta de 12 filas en el resultado del ejemplo 16.7 sin el operador CUBE a 36 filas en el resultado del ejemplo 16.8 con el operador CUBE. Para tres columnas de agrupación con valores únicos *M, N* y *P,* el número máximo de filas de subtotal producidas por el operador CUBE es $M + N + P + M * N + M * P + N * P + 1$. Como el número de filas de subtotal crece sustancialmente con el número de columnas agrupadas y los valores únicos por columna, debería utilizarse muy poco el operador CUBE cuando hay más de tres columnas agrupadas.

EJEMPLO 16.7 **Cláusula GROUP BY con tres columnas de agrupación y el resultado parcial sin subtotales**

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
      AND Sales.TimeNo = TimeDim.TimeNo
      AND (StoreNation = 'USA' OR StoreNation = 'Canada')
      AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY StoreZip, TimeMonth, TimeYear;
```

StoreZip	TimeMonth	TimeYear	SumSales
80111	1	2005	10000
80111	2	2005	12000
80111	3	2005	11000
80112	1	2005	9000
80112	2	2005	11000
80112	3	2005	15000
80111	1	2006	11000
80111	2	2006	13000
80111	3	2006	12000
80112	1	2006	10000
80112	2	2006	12000
80112	3	2006	16000

EJEMPLO 16.8 (Oracle) **Cláusula GROUP BY con tres columnas de agrupación y el resultado con subtotales producidos por el operador CUBE**

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY CUBE(StoreZip, TimeMonth, TimeYear);
```

StoreZip	TimeMonth	TimeYear	SumSales
80111	1	2005	10000
80111	2	2005	12000
80111	3	2005	11000
80112	1	2005	9000
80112	2	2005	11000
80112	3	2005	15000
80111	1	2006	11000
80111	2	2006	13000
80111	3	2006	12000
80112	1	2006	10000
80112	2	2006	12000
80112	3	2006	16000
80111	1		21000
80111	2		25000
80111	3		23000
80112	1		19000
80112	2		22000
80112	3		31000
80111		2005	33000
80111		2006	36000
80112		2005	35000

StoreZip	TimeMonth	TimeYear	SumSales
80112		2006	38000
	1	2005	19000
	2	2005	23000
	3	2005	26000
	1	2006	21000
	2	2006	25000
	3	2006	28000
80111			69000
80112			73000
	1		40000
	2		48000
	3		54000
		2005	68000
		2006	74000
			142000

El operador CUBE no es un operador primitivo. El resultado de una operación CUBE puede producirse usando varios enunciados SELECT conectados por medio del operador UNION, como se muestra en el ejemplo 16.9. Los enunciados SELECT adicionales generan subtotales para cada combinación de columnas agrupadas. Con dos columnas agrupadas, se necesitan tres enunciados SELECT adicionales para generar los subtotales. Con N columnas agrupadas, son necesarios $2^N - 1$ enunciados SELECT adicionales. Obviamente, es mucho más fácil escribir el operador CUBE que un gran número de enunciados SELECT.

EJEMPLO 16.9 **Reescritura del ejemplo 16.6 sin usar el operador CUBE**

En cada enunciado SELECT adicional, un valor predeterminado (0 para las columnas numéricas y " para columnas de texto) reemplaza la columna en donde no se generan totales.

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear = 2005
  GROUP BY StoreZip, TimeMonth
  UNION
SELECT StoreZip, 0, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear = 2005
  GROUP BY StoreZip
UNION
SELECT '', TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
```

```
        AND Sales.TimeNo = TimeDim.TimeNo
        AND (StoreNation = 'USA' OR StoreNation = 'Canada')
        AND TimeYear = 2005
  GROUP BY TimeMonth
UNION
SELECT '', 0, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
        AND Sales.TimeNo = TimeDim.TimeNo
        AND (StoreNation = 'USA' OR StoreNation = 'Canada')
        AND TimeYear = 2005;
```

operador ROLLUP operador que aumenta el resultado normal de GROUP BY con un conjunto parcial de subtotales. El operador ROLLUP es apropiado para resumir niveles de una jerarquía de dimensión.

Operador ROLLUP

El operador SQL ROLLUP proporciona una capacidad similar al operador de ascenso para cubos de datos. El operador de ascenso para cubos de datos produce totales para más partes generales de una jerarquía de dimensión. El operador SQL ROLLUP produce subtotales para cada subconjunto ordenado de columnas agrupadas para simular los efectos del operador de ascenso para cubos de datos. Por ejemplo, la operación de ***SQL ROLLUP (TimeYear, TimeQuarter, TimeMonth, TimeDay)*** produce subtotales para los subconjuntos de columna *<TimeYear, TimeQuarter, TimeMonth>*, *<TimeYear, TimeQuarter>*, *<TimeYear>*, así como el gran total. Como implica este ejemplo, el orden de las columnas en una operación ROLLUP es significativo.

Como indica el párrafo anterior, el operador ROLLUP produce sólo un conjunto parcial de subtotales para las columnas en una cláusula GROUP BY. Los ejemplos 16.10 y 16.11 demuestran el operador ROLLUP con el fin de contrastarlo con el operador CUBE en el ejemplo 16.6.

EJEMPLO 16.10 (Oracle) **Clásusula GROUP BY y resultado con subtotales producidos por el operador ROLLUP**

This example should be compared with Example 16.6 to understand the difference between the CUBE and ROLLUP operators.

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
        AND Sales.TimeNo = TimeDim.TimeNo
        AND (StoreNation = 'USA' OR StoreNation = 'Canada')
        AND TimeYear = 2005
  GROUP BY ROLLUP(StoreZip, TimeMonth);
```

StoreZip	TimeMonth	SumSales
80111	1	10000
80111	2	12000
80111	3	11000
80112	1	9000
80112	2	11000
80112	3	15000
80111		33000
80112		35000
		68000

Nótese que el ejemplo 16.10 contiene tres filas subtotales comparadas con seis filas subtotales. Este ejemplo debe compararse con el ejemplo 16.6 para entender la diferencia entre los operadores CUBE y ROLLUP en el ejemplo 16.6 con el operador CUBE. En el ejemplo 16.11, se producen subtotales para los valores en las combinaciones de columna <*StoreZip, TimeMonth*>, <*StoreZip*>, y el gran total. En el ejemplo 16.8, con el operador CUBE, también se producen subtotales para los valores en las combinaciones de columna <*StoreZip, TimeYear*>, <*TimeMonth, TimeYear*>, <*TimeMonth*> y <*TimeYear*>. Por lo tanto, el operador ROLLUP produce muchas menos filas de subtotal para el operador CUBE conforme se incrementa el número de columnas agrupadas y los valores únicos por columna.

EJEMPLO 16.11 (Oracle) **Cláusula GROUP BY con tres columnas de agrupación y el resultado con subtotales producidos por el operador ROLLUP**

Este ejemplo debe compararse con el ejemplo 16.8 para entender la diferencia entre los operadores CUBE y ROLLUP.

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
      AND Sales.TimeNo = TimeDim.TimeNo
      AND (StoreNation = 'USA' OR StoreNation = 'Canada')
      AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY ROLLUP(StoreZip, TimeMonth, TimeYear);
```

StoreZip	TimeMonth	TimeYear	SumSales
80111	1	2005	10000
80111	2	2005	12000
80111	3	2005	11000
80112	1	2005	9000
80112	2	2005	11000
80112	3	2005	15000
80111	1	2006	11000
80111	2	2006	13000
80111	3	2006	12000
80112	1	2006	10000
80112	2	2006	12000
80112	3	2006	16000
80111	1		21000
80111	2		25000
80111	3		23000
80112	1		19000
80112	2		22000
80112	3		31000
80111			69000
80112			73000
			142000

Al igual que el operador CUBE, el operador ROLLUP no es un operador primitivo. El resultado de una operación ROLLUP puede producirse utilizando varios enunciados SELECT conectados por el operador UNION, como se muestra en el ejemplo 16.12. Los enunciados SELECT adicionales generan subtotales para cada subconjunto ordenado de columnas agrupadas.

Con tres columnas agrupadas se necesitan tres enunciados SELECT adicionales para generar los subtotales. Con *N* columnas agrupadas, se necesitan *N* enunciados SELECT adicionales. Es evidente que es mucho más fácil escribir el operador ROLLUP que un gran número de enunciados SELECT adicionales.

EJEMPLO 16.12 **Reescritura del ejemplo 16.11 sin usar el operador ROLLUP**

En cada enunciado SELECT adicional, un valor predeterminado (0 para las columnas numéricas y " para columnas de texto) reemplaza la columna en donde no se generan totales.

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY StoreZip, TimeMonth, TimeYear
  UNION
SELECT StoreZip, TimeMonth, 0, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY StoreZip, TimeMonth
UNION
SELECT StoreZip, 0, 0, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY StoreZip
UNION
SELECT '', 0, 0, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.No
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006;
```

operador GROUPING SETS operador en la cláusula GROUP BY que requiere especificación explícita de las combinaciones de columna. El operador GROUPING SETS es apropiado cuando se requiere control preciso de la agrupación y los subtotales.

Operador GROUPING SETS

Para una mayor flexibilidad de la que proporcionan los operadores CUBE y ROLLUP, puede emplear el operador GROUPING SETS. Cuando usa el operador GROUPING SETS, especifica explícitamente las combinaciones de columnas para las cuales necesita totales. En contraste, la especificación de subtotales es implícita en los operadores CUBE y ROLLUP. Si no se requiere control explícito, los operadores CUBE y ROLLUP dan una especificación más sucinta.

Para ilustrar el operador GROUPING SETS, los ejemplos anteriores han sido modificados usando el operador GROUPING SETS. En el ejemplo 16.13, el operador GROUPING SETS implica subtotales para las columnas *StoreZip* y *TimeMonth* junto con el gran total expresado por el paréntesis vacío. El subconjunto (*StoreZip, TimeMonth*) también debe especificarse porque todas

las combinaciones de columna tienen que especificarse en forma explícita, incluso la agrupación normal sin el operador GROUPING SETS. El ejemplo 16.14 contiene ocho combinaciones de columna para dar el mismo resultado que el ejemplo 16.8 con el CUBE de tres columnas. El ejemplo 16.15 contiene tres combinaciones de columna para dar el mismo resultado que el ejemplo 16.11 con el ROLLUP de tres columnas.

EJEMPLO 16.13 (Oracle) **Cláusula GROUP BY con el uso del operador GROUPING SETS**

Este ejemplo produce el mismo resultado que el ejemplo 16.6.

```
SELECT StoreZip, TimeMonth, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear = 2005
  GROUP BY GROUPING SETS((StoreZip, TimeMonth), StoreZip,
            TimeMonth, ());
```

EJEMPLO 16.14 (Oracle) **Cláusula GROUP BY con el uso del operador GROUPING SETS**

Este ejemplo produce el mismo resultado que el ejemplo 16.8.

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY GROUPING SETS((StoreZip, TimeMonth, TimeYear),
         (StoreZip,TimeMonth), (StoreZip, TimeYear),
         (TimeMonth, TimeYear), StoreZip, TimeMonth, TimeYear, ()    );
```

EJEMPLO 16.15 (Oracle) **Cláusula GROUP BY con el uso del operador GROUPING SETS**

Este ejemplo produce el mismo resultado que el ejemplo 16.11.

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY GROUPING SETS(  (StoreZip, TimeMonth, TimeYear),
         (StoreZip,TimeMonth), StoreZip,()    );
```

El ejemplo 16.16 ilustra una situación en donde es preferible el operador GROUPING SETS que el operador CUBE. Como las columnas *TimeYear* y *TimeMonth* son de la misma jerarquía de dimensión, por lo general no se garantiza un cubo completo. En su lugar, es posible utilizar el operador GROUPING SETS para especificar las combinaciones de columna de las que se necesitan subtotales. Los subtotales que implican *TimeMonth* y *TimeYear* se excluyen en el ejemplo 16.16, pero se incluye en una operación completa de CUBE.

EJEMPLO 16.16 (Oracle) **Cláusula GROUP BY con el uso del operador GROUPING SETS para indicar las combinaciones de columna de las que se necesitan subtotales**

```
SELECT StoreZip, TimeMonth, TimeYear, SUM(SalesDollar) AS SumSales
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND (StoreNation = 'USA' OR StoreNation = 'Canada')
       AND TimeYear BETWEEN 2005 AND 2006
  GROUP BY GROUPING SETS(  (StoreZip, TimeYear, TimeMonth),
         (StoreZip, TimeYear), (TimeMonth, TimeYear),
         StoreZip, TimeYear, ()    );
```

Variaciones de los operadores CUBE, ROLLUP y GROUPING SETS

Los operadores CUBE, ROLLUP y GROUPING SETS pueden combinarse para dar la mezcla adecuada de especificaciones de agrupación con el operador GROUPING SETS junto con subtotales proporcionados por los operadores ROLLUP y CUBE. Esta lista proporciona algunas de las variaciones posibles cuando se utilizan estos operadores.

- Puede hacerse un CUBE parcial para producir subtotales para un subconjunto de dimensiones independientes. Por ejemplo, la cláusula *GROUP BY TimeMonth, CUBE(ItemBrand, StoreState)* produce totales en los subconjuntos de columna <*TimeMonth, ItemBrand, StoreState*>, <*TimeMonth, ItemBrand*>, <*TimeMonth, StoreState*> y <*TimeMonth*>.
- Puede hacerse un ROLLUP parcial para producir subtotales de columnas de la misma jerarquía de dimensión. Por ejemplo, la cláusula *GROUP BY ItemBrand, ROLLUP(TimeYear, TimeMonth, TimeDay)* produce totales en los subconjuntos de columna <*ItemBrand, TimeYear, TimeMonth, TimeDay*>, <*ItemBrand, TimeYear, TimeMonth*>, <*ItemBrand, TimeYear*> y <*ItemBrand*>.
- Es posible usar columnas compuestas con los operadores CUBE y ROLLUP para saltarse algunos subtotales. Por ejemplo, la cláusula *GROUP BY ROLLUP (TimeYear, (TimeQuarter, Time Month), TimeDay)* produce totales en los subconjuntos de columna <*TimeYear, TimeQuarter, TimeMonth, TimeDay*>, <*TimeYear, TimeQuarter, TimeMonth*>, <*TimeYear*> y < >. La columna compuesta <*TimeQarter, TimeMonth*> se trata como una columna sencilla en la operación ROLLUP.
- Las operaciones CUBE y ROLLUP pueden incluirse en una operación GROUPING SETS. Por ejemplo, la cláusula *GROUP BY GROUPING SETS (ItemBrand, ROLLUP (TimeYear, TimeMonth), StoreState)* produce totales en los subconjuntos de columna <*ItemBrand*>, <*StoreState*>, <*TimeYear, TimeMonth*>, <*TimeYear*> y < >. La operación empaquetada ROLLUP crea subtotales en los subconjuntos de columna <*TimeYear, TimeMonth*>, <*TimeYear*> y < >.

Otras extensiones para el apoyo a las decisiones

Además de las extensiones GROUP BY, pueden usarse varias funciones agregadas en el enunciado SELECT. A continuación se listan algunas extensiones comunes.

- La función de clasificación soporta consultas para el porcentaje superior e inferior de los resultados.
- Las funciones de razón simplifican la formulación de consultas que comparan valores individuales con los totales de grupo.
- Las funciones para mover totales y promedios permiten facilitar el análisis de datos de la serie de tiempo. La extensión OLAP en SQL:2003 proporciona la cláusula WINDOW para especificar los promedios de movimiento.

16.3.4 Vistas materializadas y reescritura de consultas

vista materializada
vista almacenada que debe sincronizarse periódicamente con sus datos de origen. Las vistas materializadas soportan el almacenamiento de datos resumidos para respuesta rápida de consultas.

Para soportar las consultas que implican grandes tablas de hechos, los DBMSs relacionales proporcionan vistas materializadas. Una vista materializada es una vista almacenada que debe sincronizarse de manera periódica con sus datos de origen. Las vistas materializadas son atractivas en los data warehouse porque los datos de origen son estables para refrescamientos periódicos que generalmente se realizan durante horas no pico. En contraste, las vistas tradicionales (no materializadas) dominan el procesamiento de la base de datos operacional porque el costo de refrescamiento puede ser alto. Junto con las vistas materializadas, la mayoría de los DBMSs relacionales soportan la sustitución automática de las vistas materializadas para tablas de origen en un proceso conocido como reescritura de consultas. Esta sección ilustra las vistas materializadas utilizando la sintaxis de Oracle 10g y ofrece ejemplos del proceso de reescritura de consultas.

Vistas materializadas en Oracle 10g

La especificación de una vista materializada en Oracle 10g implica elementos de especificación de la tabla base y de la diagramación de vistas tradicionales, junto con la especificación de las propiedades de la materialización. Debido a que se almacenan las vistas materializadas, la mayor parte de las propiedades de almacenamiento para las tablas base también puede especificarse para las vistas materializadas. Puesto que aquí no nos concentramos en las propiedades de almacenamiento, no se ilustrarán. La especificación del mapeo es la misma para las vistas tradicionales que para las vistas materializadas. Un enunciado SELECT ofrece el mapeo necesario para poblar una vista materializada. Las propiedades de materialización incluyen

- **Método de refrescamiento (incremental o completo):** Oracle tiene varias restricciones sobre los tipos de vistas materializadas que pueden refrescarse incrementalmente, de modo que aquí no se estudiará el refrescamiento incremental.
- **Tiempo de refrescamiento (sobre pedido o por asignación):** Para la opción sobre pedido, Oracle proporciona el paquete DBMS_MView con varios procedimientos de refrescamiento para especificar detalles del tiempo de refrescamiento (Refresh, Refresh_All_MViews, Refresh_Dependent).
- **Tiempo de construcción (inmediato o diferido):** Para la opción diferida, los procedimientos de refrescamiento en la DBMS_MView pueden utilizarse para especificar los detalles de la población de la vista materializada.

Los ejemplos 16.17 a 16.19 ilustran la sintaxis del enunciado CREATE MATERIALIZED VIEW. Estos enunciados parecen similares a los enunciados CREATE VIEW, excepto por las cláusulas de materialización. El tiempo de construcción es inmediato en los ejemplos 16.17 y 16.19, en tanto que es diferido en el ejemplo 16.18. El método de refrescamiento es completo y el tiempo de refrescamiento es sobre pedido en las tres vistas materializadas. El enunciado SELECT después de la palabra clave AS proporciona la diagramación para poblar una vista materializada.

La conciencia del usuario es otra diferencia entre las vistas tradicionales y las vistas materializadas. Para consultas con el uso de bases de datos operacionales, se emplean vistas tradicionales en lugar de tablas base para simplificar la formulación de consultas. Un usuario percibe la base de datos como una vista que escuda al usuario de las complejidades de las tablas base. En contraste, las consultas al data warehouse solicitadas por los usuarios, se usan tablas de hechos y dimensiones. Las tablas de hechos y dimensiones pueden ocultarse por medio de una

EJEMPLO 16.17 (Oracle) **Vista materializada que contiene las ventas para todos los países para los años posteriores a 2003 agrupadas por estado y año**

```
CREATE MATERIALIZED VIEW MV1
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE AS
SELECT StoreState, TimeYear, SUM(SalesDollar) AS SUMDollar1
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND TimeYear > 2003
  GROUP BY StoreState, TimeYear;
```

EJEMPLO 16.18 (Oracle) **Vista materializada que contiene las ventas en Estados Unidos en todos los años agrupadas por estado, año y mes**

```
CREATE MATERIALIZED VIEW MV2
BUILD DEFERRED
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE AS
SELECT StoreState, TimeYear, TimeMonth,
            SUM(SalesDollar) AS SUMDollar2
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND StoreNation = 'USA'
  GROUP BY StoreState, TimeYear, TimeMonth;
```

EJEMPLO 16.19 (Oracle) **Vista materializada que contiene las ventas en Canadá antes de 2004 agrupadas por ciudad, año y mes**

```
CREATE MATERIALIZED VIEW MV3
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE AS
SELECT StoreCity, TimeYear, TimeMonth,
            SUM(SalesDollar) AS SUMDollar3
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND StoreNation = 'Canada'
       AND TimeYear <= 2003
  GROUP BY StoreCity, TimeYear, TimeMonth;
```

herramienta de consulta para simplificar la formulación de consultas. Los usuarios del data warehouse no están conscientes de las vistas materializadas, que sólo son elementos de ayuda para el desempeño administrado por el DBMS. El DBMS proporciona herramientas para ayudar

FIGURA 16.13
Flujo del proceso de modificación de vistas

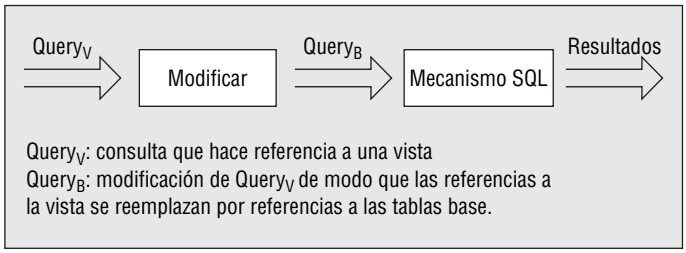
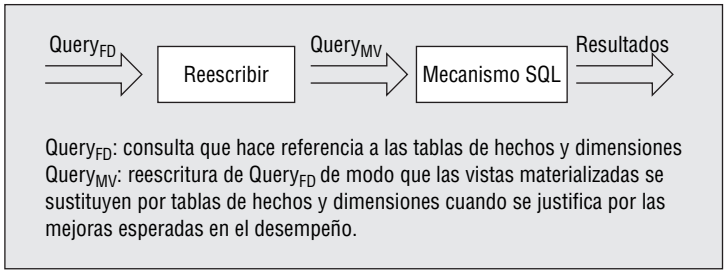


FIGURA 16.14
Flujo del Proceso de reescritura de consultas



a determinar las vistas materializadas apropiadas y usarlas con el fin de mejorar el desempeño de las consultas en un proceso que recibe el nombre de reescritura de consultas.

Principios de reescritura de consultas

reescritura de consultas
proceso de sustitución en que una vista materializada reemplaza las referencias a las tablas de hechos y dimensiones en una consulta. El optimizador de consultas evalúa si la sustitución mejorará el desempeño de la consulta original sin la sustitución de la vista materializada.

El proceso de reescritura de consultas para las vistas materializadas revierte el proceso de modificación de consultas para las vistas tradicionales que se presentan en el capítulo 10. Recuerde que el proceso de modificación de consultas (véase la figura 16.13) sustituye las tablas base por vistas, de manera que no se necesita la materialización de las vistas. En contraste, el proceso de reescritura de consultas (véase la figura 16.14) sustituye las vistas materializadas con tablas de hechos y dimensiones con el fin de evitar entrar a grandes tablas de hechos y dimensiones. Sólo se lleva a cabo el proceso de sustitución si se esperan mejoras en el desempeño.

Por lo general, la reescritura de consultas es más compleja que la modificación de consultas porque las reescritura de consultas implica un proceso de sustitución más complicado y requiere del optimizador para evaluar los costos. En ambos procesos el DBMS, y no el usuario, realiza el proceso de sustitución. En el proceso de reescritura de consultas, el optimizador debe evaluar si la sustitución mejorará el desempeño de la consulta original. En el proceso de modificación de consultas, el optimizador de consultas no compara el costo de la consulta modificada con la consulta original porque la modificación generalmente ofrece ahorros sustanciales.

Detalles de la reescritura de consultas

La reescritura de consultas implica un proceso de coincidencia entre una consulta mediante el uso de tablas de hechos y dimensiones, y una serie de vistas materializadas que contienen datos resumidos. En síntesis, una vista materializada puede proporcionar datos para una consulta si la vista materializada concuerda con las condiciones de fila, columnas de agrupación y funciones agregadas, como se explica a continuación y se resume en la tabla 16.9.

- **Coincidencia de condición de fila:** La reescritura no es posible si una vista materializada contiene más condiciones restrictivas que las condiciones de la vista materializada. Por ejemplo, si una vista materializada tiene las condiciones StoreNation = 'USA' y TimeYear = 2005, pero una consulta sólo tiene la condición StoreNation = 'USA', la vista materializada no puede proporcionar los datos para la consulta.

TABLA 16.9
Resumen de requerimientos de coincidencia para la reescritura de consultas

Tipo de coincidencia	Requerimientos
Condiciones de fila	Las condiciones de la consulta deben ser por lo menos tan restrictivas como las condiciones de las vistas materializadas.
Detalle de agrupación	Las columnas de agrupación de consultas no deben ser más detalladas que las columnas de agrupación de vista materializada.
Dependencias de agrupación	Las columnas de consulta deben coincidir o poderse derivar por dependencias funcionales de las columnas de vista materializada.
Funciones agregadas	Las funciones agregadas de consulta deben coincidir o poderse derivar de las funciones agregadas de vista materializada.

TABLA 16.10
Coincidencia de vista materializada y ejemplo 16.20

	Vista materializada	Consulta
Agrupación	StoreState, TimeYear	StoreState, TimeYear
Condiciones	TimeYear > 2003	TimeYear = 2005 StoreNation = ('USA', 'Canada')
Agregados	SUM(SalesDollar)	SUM(SalesDollar)

- **Coincidencia de agrupación para nivel de detalle:** La reescritura no es posible si la agrupación de la vista materializada tiene un nivel más alto (menos detallado) que una consulta. Desde la perspectiva de una consulta, las columnas de agrupación no deben ser más detalladas que las columnas de agrupación en una vista materializada. Por ejemplo, una consulta con una agrupación en *TimeMonth,* no puede utilizar una vista materializada con una agrupación en *TimeYear.* Sin embargo, puede ascenderse una materializada con agrupación en *TimeMonth* para proporcionar datos para una consulta con agrupación en *TimeYear.*
- **Coincidencia de agrupación para dependencias funcionales:** La reescritura no es posible si una consulta contiene columnas de agrupación que no están en una vista materializada, a menos que las columnas puedan derivarse por dependencias funcionales. Las dependencias funcionales se derivan de llaves primarias, llaves candidatas y dependencias de dimensión (vía la cláusula DETERMINES). Por ejemplo, una consulta con una agrupación en *StoreCity* puede derivarse de una vista materializada con una agrupación en *StoreId,* porque *StoreId* → *StoreCity.* Pueden usarse enlaces para recuperar columnas en una consulta, pero no en una vista materializada en tanto que haya una relación funcional (por lo general una relación 1-M) que implique tablas.
- **Coincidencia de agregado:** Los agregados en la consulta deben coincidir con los agregados disponibles en la vista materializada. Por ejemplo, una consulta que contiene un promedio puede derivarse de una vista materializada que contiene suma y cuenta.

El ejemplo 16.20 presenta un ejemplo de consulta del data warehouse y la consulta reescrita para ilustrar el proceso de coincidencia. La tabla 16.10 ilustra coincidencias entre *MV1* y la consulta del ejemplo 16.20. *MV1* y la consulta coinciden directamente en las columnas de agrupación y los cálculos agregados. La condición en *TimeYear* (> 2003) en *MV1* contiene la condición de consulta (2005). Además, la consulta contiene una condición extra en *StoreNation.* La agrupación no es necesaria en la consulta reescrita porque ya se realizó una agrupación idéntica en *MV1.*

El ejemplo 16.21 presenta un ejemplo más complejo de la reescritura de consultas que implica tres bloques SELECT combinados con el uso del operador UNION. La tabla 16.11 ilustra la coincidencia entre las vistas materializadas (*MV1, MV2, MV3*) y la consulta del ejemplo 16.21. El primer bloque de la consulta recupera el total de las ventas en Estados Unidos o Canadá, de 2003 a 2006. El segundo bloque de la consulta recupera las ventas por tienda en Estados Unidos

EJEMPLO 16.20 **Consulta de data warehouse y consulta reescrita con el uso de la vista materializada**

```
-- Consulta de data warehouse
SELECT StoreState, TimeYear, SUM(SalesDollar)
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
        AND Sales.TimeNo = TimeDim.TimeNo
        AND StoreNation IN ('USA', 'Canada')
        AND TimeYear = 2005
  GROUP BY StoreState, TimeYear;

-- Reescritura de Consulta: reemplazar tablas Sales y Time con MV1
SELECT DISTINCT MV1.StoreState, TimeYear, SumDollar1
FROM MV1, Store
WHERE MV1.StoreState = Store.StoreState
        AND TimeYear = 2005
        AND StoreNation IN ('USA','Canada');
```

en 2003. El tercer bloque de la consulta recupera las ventas por tienda en Canadá en 2003. Las cláusulas GROUP BY son necesarias en el segundo y el tercer bloque para ascender al nivel más fino de detalle de las vistas materializadas. En el tercer bloque de la consulta, la condición en *StoreNation* es necesaria porque algunas ciudades tienen nombres idénticos en ambos países.

El ejemplo 16.22 extiende el ejemplo 16.21 con un operador CUBE. En la consulta reescrita, el CUBE se realiza una vez al final, en lugar de hacerlo en cada bloque SELECT. Para ejecutar el CUBE una vez, la cláusula FROM debe contener una consulta empaquetada en un enunciado CREATE VIEW separado.

EJEMPLO 16.21 **Consulta de data warehouse y consulta reescrita con el uso de las vistas materializadas *MV1, MV2* y *MV3***

```
-- Consulta de data warehouse
SELECT StoreState, TimeYear, SUM(SalesDollar)
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
        AND Sales.TimeNo = TimeDim.TimeNo
        AND StoreNation IN ('USA','Canada')
        AND TimeYear BETWEEN 2003 and 2006
 GROUP BY StoreState, TimeYear;

-- Reescritura de consulta
SELECT DISTINCT MV1.StoreState, TimeYear, SumDollar1 AS StoreSales
FROM MV1, Store
WHERE MV1.StoreState = Store.StoreState
     AND TimeYear <= 2006
     AND StoreNation IN ('USA','Canada')
UNION
SELECT StoreState, TimeYear, SUM(SumDollar2) as StoreSales
```

```
FROM MV2
WHERE TimeYear = 2003
GROUP BY StoreState, TimeYear
UNION
SELECT DISTINCT StoreState, TimeYear, SUM(SumDollar3) as StoreSales
FROM MV3, Store
WHERE MV3.StoreCity = Store.StoreCity
      AND TimeYear = 2003 AND StoreNation = 'Canada'
GROUP BY StoreState, TimeYear;
```

TABLA 16.11
Coincidencia de vista materializadas y ejemplo 16.21

	MV1	MV2	MV3	Consulta
Agrupación	StoreState, TimeYear	StoreState, TimeMonth, TimeYear	StoreCity, StoreState, TimeYear	StoreState, TimeYear
Condiciones	TimeYear > 2003	StoreNation = 'USA'	TimeYear <= 2003 StoreNation = 'Canada'	TimeYear BETWEEN 2003 AND 2006 StoreNation = ('USA', 'Canada')
Agregados	SUM(SalesDollar)	SUM(SalesDollar)	SUM(SalesDollar)	SUM(SalesDollar)

Estos ejemplos indican el rango de posibilidades de la reescritura de consultas más que las capacidades de los DBMSs reales. La mayoría de los DBMSs empresariales soportan la reescritura de consultas, pero varía el rango de la reescritura de consultas soportada. Dada la complejidad y la naturaleza patentada de los algoritmos de reescritura de consultas, los detalles de los algoritmos de reescritura de consultas van más allá del alcance de este libro de texto.

EJEMPLO 16.22 **Consulta de data warehouse y consulta reescrita con el uso de las vistas materializadas *MV1, MV2* y *MV3***

```
-- Consulta de data warehouse
SELECT StoreState, TimeYear, SUM(SalesDollar)
  FROM Sales, Store, TimeDim
  WHERE Sales.StoreId = Store.StoreId
       AND Sales.TimeNo = TimeDim.TimeNo
       AND StoreNation IN ('USA', 'Canada')
       AND TimeYear BETWEEN 2003 and 2006
  GROUP BY CUBE(StoreState, TimeYear);

-- Reescritura de consulta
SELECT StoreState, TimeYear, SUM(StoreSales) as SumStoreSales
  FROM (
    SELECT DISTINCT MV1.StoreState, TimeYear, SumDollar1 AS StoreSales
      FROM MV1, Store
      WHERE MV1.StoreState = Store.StoreState
           AND TimeYear <= 2006
           AND StoreNation IN ('USA','Canada')
```

```
UNION
  SELECT StoreState, TimeYear, SUM(SumDollar2) as StoreSales
    FROM MV2
    WHERE TimeYear = 2003
    GROUP BY StoreState, TimeYear
UNION
  SELECT DISTINCT StoreState, TimeYear, SUM(SumDollar3) as StoreSales
    FROM MV3, Store
    WHERE MV3.StoreCity = Store.StoreCity
        AND TimeYear = 2003 AND StoreNation = 'Canada'
    GROUP BYStoreState, TimeYear    )
GROUP BY CUBE(StoreState, TimeYear);
```

16.3.5 Tecnologías de almacenamiento y optimización

Se han desarrollado varias tecnologías de almacenamiento para ofrecer capacidades de datos multidimensionales. Las tecnologías de almacenamiento soportan On Line Analytic Processing (OLAP), un nombre genérico aplicado a las capacidades de apoyo a las decisiones para los cubos de datos. Esta sección describe las características de las tecnologías de almacenamiento OLAP con un énfasis en las actuales tendencias del mercado.

MOLAP (OLAP Multidimensional)

MOLAP
mecanismo de almacenamiento que almacena y manipula directamente cubos de datos. Los mecanismos MOLAP generalmente ofrecen el mejor desempeño de la consulta, pero ponen límites para el tamaño de los cubos de datos.

En un principio, los fabricantes de software de apoyo a las decisiones desarrollaron una arquitectura de almacenamiento que manipulaba los cubos de datos de manera directa. Esta arquitectura de almacenamiento, conocida como MOLAP, por OLAP Multidimensional, era la única opción como tecnología de almacenamiento para los data warehouse hasta mediados de la década de los noventa. En la actualidad, MOLAP se ha visto eclipsado como la principal arquitectura de almacenamiento para los data warehouse, pero sigue siendo una tecnología importante para los cubos de datos de resumen y pequeños data warehouse y data mart.

Los mecanismos de almacenamiento MOLAP manipulan directamente los cubos de datos almacenados. Los mecanismos de almacenamiento de los sistemas MOLAP se optimizan para las características únicas de los datos multidimensionales, como esparcimiento y agregación compleja en miles de celdas. Como los cubos de datos se precalculan, el desempeño de la consulta MOLAP por lo general es mejor que los planteamientos competidores que usan el almacenamiento de base de datos relacional. Aun con técnicas para manejar el esparcimiento, los mecanismos de MOLAP pueden verse abrumados por el tamaño de los cubos de datos. Un cubo de datos calculado por completo puede expandirse muchas veces en comparación con los datos de entrada en bruto. Este problema de explosión de datos limita el tamaño de los cubos de datos que los mecanismos MOLAP pueden manipular.

ROLAP (OLAP Relacional)

ROLAP
extensiones del DBMS relacional para soportar datos multidimensionales. Los mecanismos ROLAP soportan una variedad de técnicas de almacenamiento y optimización para la recuperación de datos de resumen.

Gracias al tamaño del mercado potencial y al crecimiento del procesamiento de data warehouse, los fabricantes de DBMS relacionales han extendido sus productos con características adicionales para soportar operaciones y estructuras de almacenamiento para datos multidimensionales. Estas extensiones de los productos se conocen colectivamente como ROLAP, por OLAP Relacional. Debido al tamaño creciente de los data warehouse y a su investigación y desarrollo intensivos por parte de fabricantes de DBMS, fue sólo cuestión de tiempo para que el ROLAP dominara el mecanismo de almacenamiento para los data warehouse.

En el planteamiento del ROLAP, las bases de datos relacionales almacenan datos multidimensionales con el uso del esquema de estrella o sus variaciones, como se describió en la sección 16.3.1. Los cubos de datos se construyen dinámicamente a partir de tablas de hechos y

dimensiones, así como de vistas materializadas. Por lo regular, sólo debe construirse un subconjunto de datos, como lo especifica la consulta de un usuario. Las extensiones para SQL permiten a los usuarios manipular las dimensiones y medidas en cubos de datos virtuales, como se describió en la sección 16.3.3.

Los mecanismos ROLAP incorporan una variedad de técnicas de almacenamiento y optimización para la recuperación de datos de resumen. Esta lista explica las técnicas más sobresalientes.

- **Índices de enlace de mapa de bits** (véase el capítulo 8 para más detalles) son particularmente útiles para las columnas en las tablas de dimensión con unos cuantos valores, como *CustState.* Un índice de enlace de mapa de bits proporciona un enlace precalculado de los valores de una tabla de dimensión para las filas de una tabla de hechos. Para soportar los esquemas de copo de nieve, algunos fabricantes de DBMS soportan índices de enlace de mapa de bits para tablas de dimensión relacionadas con otras tablas de dimensión. Por ejemplo, un índice de mapa de bits para la columna *Division.DivManager* tiene un registro de índice que contiene un mapa de bits para filas relacionadas de la tabla *Store,* y un segundo mapa de bits para filas de la tabla *Sales* relacionadas con las filas de coincidencia de la tabla *Store.*
- **Optimización de consultas con enlace de estrella** utiliza índices de enlace de mapa de bits en tablas de dimensiones para reducir el número de filas en la tabla de hechos que debe recuperarse. Un enlace de estrella implica una tabla de hechos enlazada con una o más tablas de dimensión. La optimización de enlace de estrella incluye tres fases. En la primera fase, los índices de enlace de mapa de bits en cada tabla de dimensión se combinan usando el operador de unión para condiciones conectadas por el operador OR, y el operador de intersección para condiciones conectadas por el operador AND. En la segunda fase, los mapas de bits que resultan de la primera fase se combinan utilizando el operador de intersección. En la tercera fase, las filas de la tabla de hechos se recuperan con el uso del mapa de bits resultante de la segunda fase. La optimización de enlace de estrella puede dar como resultado un tiempo de ejecución sustancialmente menor en comparación con los algoritmos de enlace tradicionales que combinan dos tablas al mismo tiempo.
- **Reescritura de consultas usando vistas materializadas** pueden eliminar la necesidad de tener acceso a grandes tablas de hechos y dimensión. Si las vistas materializadas son grandes, pueden indexarse para mejorar el desempeño de la recuperación. La reescritura de consultas utiliza el optimizador de consultas para evaluar el beneficio de emplear vistas materializadas en comparación con las tablas de hechos y dimensión.
- **Asesores de almacenamiento de resumen** determinan el mejor conjunto de vistas materializadas que debería crearse y mantenerse para una carga de trabajo de consulta dada. Para que haya consistencia con otros componentes, el asesor de resumen se integra con el componente de reescritura de consultas y el optimizador de consultas.
- **Partición, simplificación y ejecución paralela de consultas** ofrecen oportunidades para disminuir el tiempo de ejecución de las consultas al data warehouse. Es necesario estudiar con cuidado las elecciones, de modo que el uso de la partición y simplificación soporten el nivel deseado de ejecución paralela de consultas.

HOLAP
mecanismo de almacenamiento para data warehouse que combina mecanismos de almacenamiento ROLAP y MOLAP. El HOLAP implica tanto almacenamiento de datos relacionales y multidimensionales, como combinación de datos de fuentes relacionales y multidimensionales para operaciones de cubo de datos.

A pesar de la investigación y el desarrollo intensivos sobre técnicas de almacenamiento y optimización ROLAP, los mecanismos MOLAP darán un mejor tiempo de respuesta a las consultas. No obstante, el almacenamiento MOLAP sufre de limitaciones en el tamaño del cubo de datos, de manera que el almacenamiento ROLAP es preferible para data warehouse detallados. Además, la diferencia en el tiempo de respuesta se ha estrechado tanto que el almacenamiento ROLAP puede implicar sólo una ligera desventaja en el desempeño si se emplean en forma adecuada las técnicas de almacenamiento y optimización ROLAP.

HOLAP (OLAP híbrido)

Debido al equilibrio entre MOLAP y ROLAP, se ha desarrollado una tercera tecnología conocida como HOLAP, por OLAP híbrido, para combinar ROLAP y MOLAP. El HOLAP permite que un data warehouse se divida entre el almacenamiento relacional de tablas de hechos y

dimensiones, y el almacenamiento multidimensional de cubos de datos de resumen. Cuando se presenta una consulta OLAP, el sistema HOLAP puede combinar los datos administrados por ROLAP y MOLAP.

A pesar del atractivo del HOLAP, tiene desventajas potenciales que pueden limitar su uso. Primero, HOLAP puede ser más complejo que ROLAP o MOLAP, en especial si un fabricante de DBMS no ofrece soporte total para HOLAP. Para poder soportar por completo HOLAP, un fabricante de DBMS debe proporcionar mecanismos tanto MOLAP como ROLAP, así como herramientas para combinar ambos mecanismos en el diseño y operación de un data warehouse. Segundo, hay una transposición considerable en la funcionalidad entre las técnicas de almacenamiento y la optimización en los mecanismos ROLAP y MOLAP. No está claro si las técnicas de almacenamiento y optimización ROLAP deben eliminarse o usarse además de las técnicas MOLAP. Tercero, debido a que la diferencia en el tiempo de respuesta se ha reducido entre ROLAP y MOLAP, su combinación quizá no ofrezca una mejora considerable en el desempeño como para justificar la complejidad mayor.

16.4 Mantenimiento de un data warehouse

Aunque los data warehouse contienen en gran medida datos reproducidos, el mantenimiento de un data warehouse es mucho más difícil que sólo copiar desde los datos de origen. Mantener un data warehouse implica inicialmente poblar el almacén con datos de origen y refrescarlo en forma periódica conforme cambian los datos de origen. La determinación para cargar en un almacén implica hacer que las necesidades de apoyo a las decisiones coincidan con las realidades de los datos disponibles. Para satisfacer las necesidades de apoyo a las decisiones, los data warehouse usan datos de muchas fuentes, tanto internas como externas. La reconciliación de las diferencias entre las fuentes de datos es un desafío significativo, en especial si consideramos que por lo general los sistemas de origen no pueden cambiarse. Conforme cambian sus datos de origen, el data warehouse debe refrescarse de modo oportuno para soportar la toma de decisiones. La determinación del tiempo y contenido que se debe refrescar puede ser un reto importante debido a que las fuentes de datos cambian con distintos índices. Como consecuencia de estos desafíos, el mantenimiento de un data warehouse es tal vez la actividad de mayor importancia en el desarrollo de un data warehouse.

Esta sección presenta varios aspectos importantes del mantenimiento de un data warehouse. La primera parte describe las clases de datos de origen disponibles para poblar un data warehouse. La segunda parte describe el flujo de trabajo para mantener un almacén, y la parte final analiza el problema que representa determinar la frecuencia del refrescamiento y el contenido que se debe refrescar.

16.4.1 Fuentes de datos

El acceso a los datos de origen presenta desafíos en el manejo de una variedad de formatos y restricciones de los sistemas de origen. Por lo regular, no es posible cambiar los sistemas de origen externo. Los sistemas de origen interno pueden o no cambiarse para ajustarse a los requerimientos de un data warehouse. Aun cuando pueda cambiarse un sistema de origen, es probable que las limitaciones presupuestales sólo permitan cambios menores. Los datos de origen pueden almacenarse en formato heredado o formato moderno. El formato heredado generalmente evita la recuperación usando lenguajes no procedurales como SQL. El formato moderno significa que puede tenerse acceso a los datos de origen como una base de datos relacional o como páginas web. Las páginas web pueden ser difíciles de aparearse y no podrán estandarizarse de un sitio web a otro a menos que se almacenen con metadatos formales.

El cambio de datos desde los sistemas de origen proporciona la base para actualizar un data warehouse. El cambio de datos compromete el nuevo origen de datos (inserciones) y modificaciones al origen de datos existente (actualizaciones o eliminaciones). Además, el cambio de datos puede afectar tablas de hechos y/o tablas de dimensiones. Los cambios de datos más comunes implican inserciones de hechos nuevos. Las inserciones de nuevas dimensiones y modificaciones de dimensiones son menos comunes, pero su captura sigue siendo importante.

TABLA 16.12
Resumen de la clasificación del cambio de datos

Tipo de cambio	Descripción	Evaluación
Cooperativo	Notificación del sistema de origen con el uso de disparadores	Requiere modificaciones de los sistemas de origen
Registrado	Actividad del sistema de origen capturada en registros	Fácilmente disponibles pero con procesamiento significativo para extraer datos útiles
Consultable	Consultas del sistema de origen usando estampas de tiempo	Requiere estampas de tiempo en los datos de origen y sistemas de origen no heredados
Fotografía instantánea	Descarga repentina de datos de origen aumentados con operaciones de diferencia	Uso intensivo de recursos para su creación y procesamiento significativo para operaciones de diferencia; ningún requerimiento de sistema de origen tan útil para los datos heredados

De acuerdo con las características del sistema de origen, los datos de cambio pueden clasificarse como de cooperación, registrados, consultables o de fotografía instantánea, como se resume en la tabla 16.12. El cambio de datos cooperativo implica la notificación del sistema de origen acerca de los cambios. La notificación por lo general ocurre con el uso de un disparador en el momento en que termina la transacción. El cambio de datos cooperativo puede capturarse de inmediato en un data warehouse o ponerse en una cola para su posible entrada posterior con otros cambios. El cambio de datos cooperativo es el formato menos común para el cambio de datos porque implica modificaciones tanto al sistema de origen como al data warehouse.

El cambio de datos registrados implica archivos que registran cambios u otra actividad del usuario. Por ejemplo, un registro de transacción contiene todos los cambios realizados por una transacción y un registro web contiene historiales de acceso a páginas por parte de los visitantes de sitios web (llamado flujo de clics). El cambio de datos registrados normalmente no implica cambios en un sistema de origen, puesto que ya están disponibles como registros para la mayor parte de los recursos del sistema. Sin embargo, los registros pueden contener grandes cantidades de datos irrelevantes. Además, la derivación de los datos relevantes necesarios para el data warehouse puede requerir la comparación del registro de sucesos relacionados, una tarea de recursos intensivos. El cambio de datos registrados se emplea con mayor frecuencia para el subconjunto de relaciones con los clientes de un data warehouse.

Como su nombre lo indica, el cambio de datos consultables proviene de una fuente de datos a través de una consulta. El cambio de datos consultables requiere del estampado de tiempo en la fuente de datos. Debido a que unas cuantas fuentes de datos contienen estampas de tiempo para todos los datos, el cambio de datos consultables por lo regular se aumenta con otra clase de datos. El cambio de datos consultables aplica más en tablas de hechos con el uso de campos como fecha de pedido, fecha de embarque y fecha de contratación, que se almacenan como fuente de datos operativos.

El cambio de datos de fotografía instantánea implica descargas repentinas periódicas de los datos de origen. Para derivar los datos de cambio, una operación de diferencia utiliza las dos fotografías instantáneas más recientes. El resultado de una operación de diferencia se conoce como delta. Las fotografías instantáneas son la forma más común de cambio de datos como consecuencia de la aplicabilidad. Las fotografías instantáneas son la única forma de cambio de datos sin requerimientos en un sistema de origen. Calcular una fotografía instantánea puede ocupar recursos intensivos, por lo que puede haber restricciones acerca del tiempo y la frecuencia de recuperación de una fotografía instantánea.

16.4.2 Flujo de trabajo para el mantenimiento de un data warehouse

Mantener un data warehouse implica una variedad de tareas que manipulan el cambio de datos de los sistemas de origen. La figura 16.15 presenta un flujo de trabajo genérico que organiza las

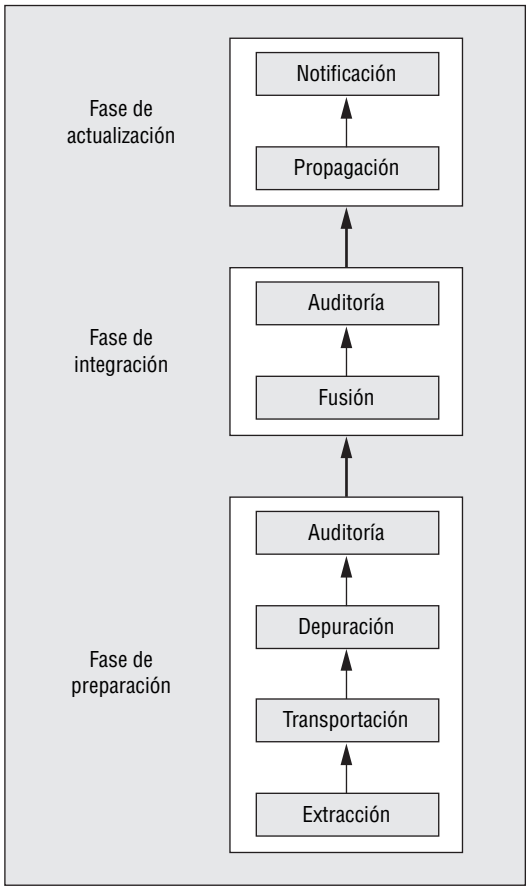


FIGURA 16.15
Flujo de trabajo genérico para el mantenimiento del data warehouse

tareas. La fase de preparación manipula el cambio de datos de sistemas de origen individual. La extracción implica la recuperación de datos de un sistema de origen individual. La transportación implica el movimiento de los datos extraídos a un área de estancia. La depuración implica una variedad de tareas para estandarizar y mejorar la calidad de los datos extraídos. La auditoría implica registrar los resultados del proceso de depuración, hacer revisiones sobre la integridad y racionalidad y manejar excepciones.

La fase de integración implica fusionar las fuentes separadas y depuradas. La fusión puede implicar la eliminación de inconsistencias entre los datos de origen. La auditoría implica registrar los resultados del proceso de fusión, efectuar revisiones de integridad y racionalidad y manejar excepciones.

La fase de actualización implica propagar el cambio de datos integrados a varias partes del data warehouse, incluidas tablas de hechos y dimensiones, vistas materializadas, cubos de datos almacenados y data marts. Después de la propagación, puede enviarse una notificación a los grupos de usuarios y administradores.

Las fases de preparación e integración deberían resolver los problemas de calidad de los datos, como se resume en la tabla 16.13. Los datos de sistemas heredados normalmente son sucios, lo que significa que pueden no conformar los estándares de calidad de datos entre ellos o entre los datos de toda la empresa. Los datos sucios pueden llevar a una toma de decisiones deficientes si se cargan directamente. La auditoría debe incluir el manejo de excepciones con el fin de resolver los problemas de calidad de los datos. Las excepciones pueden anotarse en un archivo de registro y luego manejarse manualmente. Con el paso del tiempo, las excepciones tendrían que disminuir conforme se mejoran los estándares de calidad de los datos en las fuentes de datos internas.

Además del manejo de excepciones, la tarea de auditoría debe incluir revisiones de la integridad y la racionalidad. Una revisión de la integridad cuenta el número de unidades de reporte

TABLA 16.13
Problemas comunes en la calidad de los datos

Identificadores múltiples: algunas fuentes de datos pueden utilizar diferentes llaves primarias para la misma entidad como números de cliente diferentes
Nombres múltiples: el mismo campo puede representarse usando nombres de campo distintos
Diferentes unidades: las medidas y dimensiones pueden tener diferentes unidades y granularidades
Valores faltantes: es probable que los datos no existan en algunas bases de datos; pueden utilizarse diversos valores predeterminados en las fuentes de datos para compensar los valores faltantes
Transacciones huérfanas: algunas transacciones pueden ser partes importantes perdidas, como un pedido sin cliente
Campos multipropósito: algunas bases de datos pueden combinar datos en un campo, como diferentes componentes de una dirección
Datos en conflicto: algunas fuentes de datos pueden tener datos en conflicto, como diferentes domicilios de un cliente
Diferentes horas de actualización: algunas fuentes de datos pueden hacer actualizaciones en diferentes intervalos

para asegurarse de que todas hayan reportado durante un periodo dado. Una revisión de la racionalidad determina si los hechos clave caen dentro de límites predeterminados y son una extrapolación realista de la historia previa. Las excepciones pueden requerir reconciliación por parte de los analistas de la empresa antes de la propagación al data warehouse.

El flujo de trabajo genérico de la figura 16.15 se aplica tanto a la carga inicial como al refrescamiento periódico de un almacén. La carga inicial a menudo requiere un periodo extenso de depuración de datos o resolver los problemas de la calidad de los mismos. Un objetivo de la carga inicial es descubrir los problemas de calidad de los datos y solucionarlos. El proceso de refrescamiento comúnmente varía entre fuentes de datos. El proceso del flujo de trabajo debe personalizarse para ajustarse a los requerimientos de cada fuente de datos. Por ejemplo, puede minimizarse la auditoría para las fuentes de datos de alta calidad.

herramientas ETL
herramientas de software para extracción, transformación y carga de datos desde las fuentes de datos hasta el data warehouse. Las herramientas ETL eliminan la necesidad de escribir código personalizado para muchas tareas de mantenimiento del data warehouse.

Para soportar la complejidad del mantenimiento del data warehouse, se han desarrollado productos de software que reciben el nombre de herramientas de extracción, transformación y carga (ETL). Estas herramientas, disponibles de terceros fabricantes y fabricantes de DBMS, eliminan la necesidad de escribir código personalizado para muchas tareas de mantenimiento. La mayor parte de las herramientas de ETL utilizan especificaciones de reglas en vez de codificación procedural para indicar la lógica y las acciones. Algunas herramientas de ETL pueden generar código que puede personalizarse para mayor flexibilidad. Además de las herramientas de ETL, algunos fabricantes de DBMS proporcionan programas de carga de datos y extensiones SQL patentadas que soportan las tareas de mantenimiento. Por ejemplo, Oracle 10g soporta el programa SQL Loader, así como el enunciado MERGE y el enunciado de tablas múltiples INSERT. Las herramientas de ETL y extensiones SQL patentadas son esenciales para reducir el esfuerzo de implementar las tareas del flujo de trabajo.

16.4.3 Administración del proceso de refrescamiento

Refrescar un data warehouse es un proceso complejo que implica la administración de diferencias de hora entre la actualización de fuentes de datos y la actualización de los objetos relacionados en el data warehouse (tablas, vistas materializadas, cubos de datos, data marts). En la figura 16.16, la demora válida es la diferencia entre el acontecimiento de un evento en el mundo real (hora válida) y el almacenamiento del evento en una base de datos operacional (hora de la transacción). La demora de carga es la diferencia entre la hora de la transacción y el almacenamiento del evento en un almacén (tiempo de carga). Para fuentes de datos internas, puede haber cierto control sobre la demora válida. Por lo general, no se tiene control sobre la demora válida en el caso de las fuentes de datos externas. Por consiguiente, un administrador de data warehouse debe tener control sobre la demora de carga.

La figura 16.13 implica que el origen de datos puede cambiarse de manera independiente, llevándolo a diferentes índices de cambio para las tablas de hechos y tablas de dimensiones. Las tablas de hechos generalmente registran eventos concluidos, como pedidos, embarques y compras con vínculos a dimensiones relacionadas. Por ejemplo, insertar una fila en la tabla *Sales* requiere

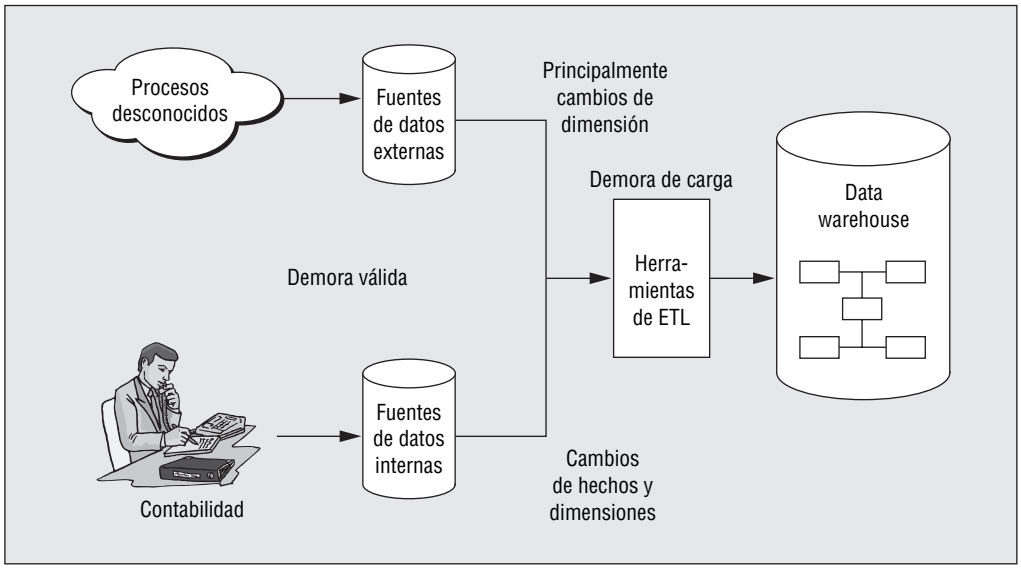


FIGURA 16.16
Panorama del proceso de refrescamiento del data warehouse

llaves foráneas que hacen referencia a las tablas *Customer, Store, TimeDim* e *Item.* Sin embargo, las actualizaciones e inserciones a las tablas de dimensión relacionadas pueden ocurrir a diferentes horas que los eventos de hechos. Por ejemplo, un cliente puede moverse o un artículo puede cambiar su precio en diferentes horas que las de los pedidos, embarques y compras de inventario. Como resultado de los diferentes índices de cambio, un administrador de data warehouse debe administrar la demora de carga por separado para las tablas de dimensiones y de hechos.

El principal objetivo de la administración del proceso de refrescamiento es determinar la frecuencia del refrescamiento para cada fuente de datos. La frecuencia de refrescamiento óptima maximiza el beneficio de refrescamiento neto, definido como el valor de las líneas de tiempo de los datos menos el costo del refrescamiento. El valor de la línea de tiempo de los datos depende de la sensibilidad de la toma de decisiones para la vigencia de los datos. Algunas decisiones son muy sensibles al tiempo, como las decisiones de inventario. Las organizaciones de una cadena de abastecimiento intentan minimizar los costos implícitos de inventario al almacenar bienes tan cerca como sea posible al momento de necesitarlos. Otras decisiones no son tan sensibles al tiempo. Por ejemplo, la decisión de cerrar una tienda con desempeño deficiente por lo regular se tomaría usando datos de un extenso periodo.

Fisher y Berndt (2001) han propuesto un método para medir el valor de la línea de tiempo de los datos. Definieron una medida del índice de error de un data warehouse con respecto a una carga de trabajo de consultas dada. Para determinar el índice de error, se estima la volatilidad de los datos de las consultas y dimensiones. Aun cuando un data warehouse almacena datos de nivel individual, la mayoría de las consultas implicarán adiciones, no recuperación de datos de nivel individual. Por ejemplo, la mayor parte de las consultas en el esquema de ventas del almacén implicarían la marca del artículo o su categoría, no los artículos individuales. Dada una evaluación del índice de error, puede tenerse acceso al valor de la línea de tiempo de los datos usando la frecuencia u otra ponderación para las consultas al data warehouse.

El costo de refrescar un data warehouse incluye tanto recursos de computación como humanos. Los recursos de computación son necesarios para todas las tareas en el flujo de trabajo del mantenimiento. Los recursos humanos pueden ser necesarios en las tareas de auditoría durante las fases de preparación e integración. El nivel de la calidad de los datos en los datos de origen también afecta el nivel de los recursos humanos requeridos. El esfuerzo de desarrollo para usar herramientas de ETL y escribir software personalizado no es parte del costo de refrescamiento, a menos que haya un costo de desarrollo en curso con cada refrescamiento. Una distinción importante implica el costo fijo y el costo variable del refrescamiento. Un alto costo fijo alienta el refrescamiento menos frecuente, porque el costo fijo ocurre con cada refrescamiento. El costo fijo puede incluir un esfuerzo de inicio y cierre, así como renta de recursos.

TABLA 16.14
Resumen de las limitaciones de refrescamiento

Tipo de limitación	Descripción
Acceso de origen	Restricciones en el tiempo y frecuencia de la extracción de cambio de datos
Integración	Restricciones que requieren reconciliación actual del cambio de datos
Integridad/consistencia	Restricciones que requieren carga de datos de cambio en el mismo periodo de refrescamiento
Disponibilidad	Restricciones de programación de carga como resultado de los aspectos de recursos incluida capacidad de almacenamiento, disponibilidad en línea y uso del servidor

El administrador del data warehouse debe satisfacer las restricciones sobre el proceso de refrescamiento junto con la cuadratura del valor de la línea de tiempo contra el costo del refrescamiento, como se resume en la tabla 16.14. Las restricciones, ya sea sobre un data warehouse o un sistema de origen, pueden limitar el refrescamiento frecuente. Las restricciones de acceso a la fuente pueden ser consecuencia de una tecnología heredada con escalabilidad restringida para fuentes de datos internas o problemas de coordinación para fuentes de datos externas. Las restricciones de integración a menudo implican identificación de entidades comunes como clientes y transacciones entre fuentes de datos. Las restricciones de integridad/consistencia pueden implicar mantenimiento del mismo periodo de tiempo en el cambio de datos o la inclusión del cambio de datos de cada fuente de datos para la integridad. La disponibilidad del data warehouse con frecuencia implica conflictos entre la disponibilidad en línea y la carga del data warehouse.

Reflexión final

Este capítulo presentó una introducción a los conceptos y práctica del almacenamiento de datos. Este capítulo empezó por examinar las diferencias conceptuales entre bases de datos relacionales, empleadas tradicionalmente para el procesamiento de transacciones, y bases de datos multidimensionales, sugeridas para la nueva generación de aplicaciones de apoyo a las decisiones. Se describieron las características únicas de datos de apoyo a las decisiones, seguido de un análisis de las arquitecturas de data warehouse, minería de datos y uso del data warehouse en las organizaciones.

Puede implementarse data warehouse usando el modelo de datos multidimensionales, el modelo relacional o una combinación de ambos modelos. Para el modelo de datos multidimensionales, este capítulo presentó la terminología asociada con los cubos de datos y operadores para manipular cubos de datos. Para el modelo relacional de datos, el capítulo presentó técnicas de modelado de datos (el esquema de estrella y sus variaciones), extensiones SQL en la cláusula GROUP BY para los datos dimensionales de consulta, vistas materializadas y tecnologías de almacenamiento. Las técnicas de almacenamiento soportan data warehouse con el uso tanto de mecanismos de relación como de almacenamiento de cubos de datos.

No obstante el modelo de datos y la arquitectura de almacenamiento, mantener un data warehouse es un proceso difícil que debe manejarse con cuidado. El capítulo presentó los tipos de fuentes de datos usadas en el mantenimiento de un data warehouse, un flujo de trabajo genérico que describe las tareas involucradas en el mantenimiento de un data warehouse, así como aspectos de diseño a considerar en el proceso de refrescamiento. El capítulo propuso el uso de las herramientas de ETL para reducir la cantidad de código de personalización para implementar los procedimientos que pueblan un data warehouse.

Revisión de conceptos

- Necesidades de datos para el procesamiento de transacciones contra aplicaciones de apoyo a las decisiones.
- Características de un data warehouse: orientado a los sujetos, integrado, variable con el tiempo y no volátil.
- Arquitecturas para desplegar un data warehouse: dos niveles, tres niveles y ascendente.

- Etapas del desarrollo de un data warehouse (prenatal, infancia, niñez, adolescencia, edad adulta y sabiduría) y dificultad de moverse entre etapas (infancia a niñez y adolescencia a edad adulta).
- Cubo de datos multidimensionales: dimensiones, medidas, jerarquías, tipo de datos de serie de tiempo.
- Operadores multidimensionales: rebanada, dado, descenso, ascenso, pivote.
- Esquema de estrella: tabla de hechos y tablas de dimensiones relacionadas.
- Variaciones del esquema de estrella: esquema de copo de nieve (niveles de dimensión múltiple) y esquema de constelación (múltiples tablas de hechos).
- Mantenimiento de la integridad dimensional histórica con el uso de una representación tipo II para historial ilimitado y una representación tipo III para historial limitado.
- Representación de dimensiones para soportar las operaciones de cubo de datos y las técnicas de reescritura de consultas.
- Extensiones de la cláusula GROUP BY para cálculo de subtotales: operadores CUBE, ROLLUP y GROUPING SETS.
- Vistas materializadas para almacenamiento de datos de resumen precalculados.
- Reescritura de consultas que implican sustitución de vistas materializadas para tablas de hechos y dimensiones con el fin de mejorar el desempeño de las consultas al data warehouse.
- Arquitectura de almacenamiento multidimensional: ROLAP, MOLAP y HOLAP.
- Tipos de cambio de datos usados para poblar un data warehouse: cooperativo, registrado, consultables y de fotografía instantánea.
- Fases del flujo de trabajo para mantener un almacén: preparación, integración y propagación.
- Importancia de las herramientas de ETL (extracción, transformación y carga) para reducir la codificación en los procedimientos para poblar un data warehouse.
- Determinación de la frecuencia de refrescamiento para un data warehouse: equilibrio de la frecuencia del refrescamiento contra el costo del refrescamiento mientras se satisfacen las restricciones de refrescamiento.
- Tipos de restricciones de refrescamiento: acceso a fuente, integración, integridad/consistencia, disponibilidad.

Preguntas

1. ¿Por qué las bases de datos operacionales no son particularmente apropiadas para las aplicaciones de apoyo a las decisiones?
2. ¿En qué difiere un data warehouse de un mercado de datos?
3. ¿Cuándo es más adecuada la arquitectura de data warehouse de tres niveles que la de dos niveles?
4. ¿Cuáles son los componentes de un modelo de datos empresariales?
5. ¿Cuáles son algunas causas de los fracasos en los proyectos de data warehouse?
6. ¿Una arquitectura ascendente de data warehouse utiliza un modelo de datos empresariales?
7. ¿Qué es un mercado operativo?
8. ¿Cuál es el propósito del modelo de madurez del data warehouse?
9. ¿Qué discernimiento importante ofrece el modelo de madurez del data warehouse?
10. ¿Cuáles son las ventajas de la representación multidimensional sobre la representación relacional para los analistas empresariales?
11. Explique por qué una dimensión puede tener múltiples jerarquías.
12. ¿Cuáles son las ventajas del uso de datos de serie de tiempo en una celda en vez del tiempo como una dimensión?
13. ¿En qué difiere el rebanado de un cubo de datos del manejo de dados?
14. ¿Cuáles son las diferencias entre el descenso y el ascenso de una dimensión de cubo de datos?