
Diseño e implementación multidimensional de un *Data Warehouse*

PID_00236072

Alberto Abelló Gamazo
Carles Llorach Rius



Los textos e imágenes publicados en esta obra están sujetos –excepto que se indique lo contrario– a una licencia de Reconocimiento-NoComercial-SinObraDerivada (BY-NC-ND) v.3.0 España de Creative Commons. Podéis copiarlos, distribuirlos y transmitirlos públicamente siempre que citéis el autor y la fuente (FUOC. Fundación para la Universitat Oberta de Catalunya), no hagáis de ellos un uso comercial y ni obra derivada. La licencia completa se puede consultar en <http://creativecommons.org/licenses/by-nc-nd/3.0/es/legalcode.es>

Índice

Introducción.....	5
Objetivos.....	7
1. Las necesidades de los analistas y las herramientas OLAP.....	9
1.1. Bases de datos estadísticas	10
1.2. Hojas de cálculo	11
1.3. Herramientas OLAP y multidimensionalidad	12
1.4. Modelización de un almacén de datos y multidimensionalidad	21
2. Componentes del modelo multidimensional.....	25
2.1. Estructuras de datos	25
2.1.1. Dimensiones	26
2.1.2. Hechos	29
2.2. Operaciones sobre los datos	34
2.3. Restricciones de integridad inherentes al modelo	39
2.3.1. Unicidad y entidad de la Base	39
2.3.2. Acumulación o agregación	40
2.3.3. Transitividad	44
3. Diseño conceptual.....	45
3.1. Una metodología para diseñar una Estrella	45
3.1.1. Elegir el Hecho	46
3.1.2. Encontrar el gránulo oportuno	46
3.1.3. Elegir las dimensiones que se utilizarán en el análisis ...	47
3.1.4. Encontrar los atributos de cada dimensión	48
3.1.5. Distinguir entre descriptores y jerarquías de agregación	50
3.1.6. Decidir cuáles son las medidas que interesan	51
3.1.7. Definir Celdas	52
3.1.8. Explicitar las restricciones de integridad	53
3.1.9. Estudiar la viabilidad	54
3.2. Reconsideraciones en el diseño conceptual	55
3.2.1. Dimensiones con múltiples roles	55
3.2.2. Dependencias entre dimensiones	56
3.2.3. Minidimensiones	58
3.2.4. Diseño de datos heterogéneos	58
3.2.5. Hechos con una sola medida	60
4. Diseño lógico.....	62

4.1. La Estrella (el caso básico)	62
4.2. El copo de nieve	64
4.3. Conformación: partición de dimensiones	66
4.4. Dimensiones degeneradas, de desechos y sombras	68
4.5. Generalizaciones/especializaciones	69
4.6. Estructuras temporales	71
5. Consultas con SQL'99.....	75
5.1. Estructura básica de la consulta	75
5.2. GROUPING SETS	77
5.2.1. ROLLUP	81
5.2.2. CUBE	83
6. Diseño físico.....	85
6.1. Plan y técnicas básicas de acceso	85
6.2. Índices de mapas de bits	87
6.3. Particiones horizontales	89
6.4. Particiones verticales, herramientas VOLAP	90
6.5. Matrices n -dimensionales, herramientas MOLAP y HOLAP	91
6.6. Técnicas de preagregación	94
7. Ejemplo de diseño multidimensional.....	98
7.1. Diseño conceptual	98
7.2. Diseño lógico	100
7.3. Diseño físico	101
7.4. Definición del cubo OLAP en Mondrian	103
8. Consultas con MDX.....	105
8.1. Estructura básica de la consulta	106
8.2. Tipos de elementos	106
8.3. Funciones básicas	108
8.4. Consultas simples	110
9. Nuevas tendencias.....	114
Resumen.....	115
Actividades.....	117
Ejercicios de autoevaluación.....	117
Solucionario.....	119
Glosario.....	122
Bibliografía.....	124

Introducción

No basta con tener un almacén de datos para disponer de toda la información que nos hace falta y ser capaces de utilizarla para tomar decisiones. La gran cantidad de datos y la falta de especialización en informática por parte de los analistas hacen que resulte imprescindible disponer de algún tipo de herramienta que facilite su consulta.

De este modo, a partir del almacén de datos corporativo, se suelen diseñar pequeños almacenes de datos departamentales que acercan los datos a los usuarios. Estos almacenes se diseñan utilizando el modelo multidimensional, de modo que se puedan utilizar herramientas OLAP para consultarlos. Las herramientas OLAP y los almacenes de datos no se excluyen, se complementan.

A grandes rasgos, el modelo multidimensional distingue dos tipos de datos: los hechos que queremos analizar y las dimensiones que utilizamos para analizarlos. Esta división produce dos beneficios: por un lado, podemos utilizar técnicas específicas de almacenamiento y acceso de datos; y por el otro, facilitamos la comprensión de los datos de modo que los analistas son capaces de formular sus consultas casi sin ningún conocimiento informático y mediante herramientas gráficas muy intuitivas.

En este módulo encontraréis una definición formal del modelo multidimensional, así como una guía para hacer un buen diseño conceptual, lógico y físico. Para facilitar las consultas multidimensionales como parte del diseño físico se verán algunas palabras reservadas del lenguaje básico estándar SQL'99.

También se abordan las interrelaciones existentes entre el diseño multidimensional y el diseño de almacenes de datos: los puntos de coincidencia, las diferencias y las implicaciones que conlleva adoptar un determinado enfoque de diseño u otro.

Con el fin de aplicar los conocimientos adquiridos y dar un enfoque más práctico al módulo se desarrolla un ejemplo completo de diseño multidimensional: desde el modelo conceptual, pasando por el lógico y llegando hasta el diseño físico. No hay que olvidar que el modelado dimensional es una forma de organizar los datos antes de convertirlos en información útil para los usuarios del negocio. El objetivo final es que estos puedan encontrar de manera intuitiva y rápida la información que necesitan. Así mismo se incorpora una pequeña guía del lenguaje MDX (*MultiDimensional query eXpression*) con la sintaxis básica, las principales funciones que se pueden utilizar y algunas consultas de ejemplo.

Finalmente, se enumeran las últimas tendencias en OLAP, basadas principalmente en tipos de motores actuales y su uso en ámbitos específicos como *Big Data*.

Objetivos

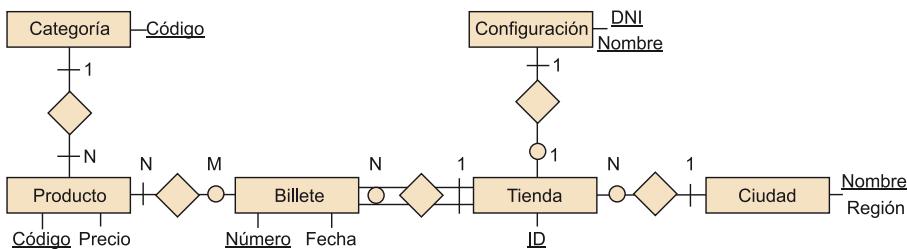
Este módulo didáctico presenta el modelo multidimensional y algunos conceptos asociados. Con este módulo, lograréis los objetivos siguientes:

- 1.** Conocer los componentes del modelo multidimensional (estructuras de datos, operaciones y restricciones de integridad).
- 2.** Entender cuál es el diseño multidimensional y los problemas de diseño que presenta (en el ámbito conceptual, así como en el lógico y físico).
- 3.** Ser capaz de diseñar una base de datos multidimensional.
- 4.** Saber qué nos ofrece el estándar SQL para facilitar las consultas multidimensionales.
- 5.** Comprender los mecanismos de almacenamiento e indexación asociados a las herramientas multidimensionales (MOLAP, ROLAP, etc.).

1. Las necesidades de los analistas y las herramientas OLAP

Hasta ahora estáis acostumbrados a ver herramientas OLTP, que están basadas en la gestión de transacciones (conjuntos de operaciones de alta, baja, modificación y consulta de datos) que ayudan en el funcionamiento diario del negocio. La pregunta que os deberíais hacer es si estas herramientas son apropiadas para analizar el funcionamiento del negocio y tomar decisiones. ¿Creéis que un directivo de una compañía será capaz de hacer una consulta con SQL sobre un esquema ER o simplemente estará dispuesto a dedicar su valioso tiempo intentando hacerla?

Figura 1



Nota

El sistema que utilizaremos serán "comillas" para las instancias, "cursiva" para términos ingleses y operaciones del modelo multidimensional, tipo de letra CourierNew para los elementos de los esquemas y "mayúsculas" para los nombres de los elementos del modelo multidimensional y las operaciones. Además, en "negrita" estará la primera aparición de los términos principales que introduzcamos.

Las dificultades de consulta para los usuarios no informáticos

Un usuario no informático ni siquiera entendería un esquema ER tan sencillo como el que tenéis en la figura 1. Además, no se trata simplemente de consultar el precio de un determinado producto, sino, dada una cadena de tiendas repartidas por todo el Estado, saber cómo han evolucionado las ventas de los diferentes productos durante el mes pasado respecto al mismo mes del año anterior.

Como ya sabemos, el modelo ER fue concebido para reducir la cantidad de datos redundantes y evitar tener que modificar muchos registros al mismo tiempo al hacer un único cambio. Esto se consigue a costa de empeorar el tiempo de respuesta a las consultas (asumiendo que habrá un número elevado de actualizaciones respecto al número de consultas). La pregunta que debemos hacernos ahora es la siguiente: ¿esto sirve para los analistas? ¿Qué querrán actualizar estos usuarios? Nada, puesto que ellos solo quieren saber cómo va la empresa, no introducirle datos nuevos.

El modelo ER no facilita precisamente la consulta de los datos. De hecho, probablemente a causa de que mezcla diferentes procesos de negocio, a los usuarios se les haría difícil incluso recordar el esquema. Además, tampoco es fácil construir un software que facilite esta consulta.

El modelo ER, que resulta muy conveniente en entornos en los que se producen cambios frecuentes, no es muy adecuado para entornos de análisis en los que lo que prevalece es tener respuestas rápidas a las consultas y que el esquema sea fácil de entender. El sistema ya no tiene que ayudar a vender, comprar, producir o transportar, sino a evaluar, comparar, presupuestar, planificar, proyectar, etc.

1.1. Bases de datos estadísticas

Lo primero que se nos podría ocurrir para solucionar este problema es pensar en las bases de datos estadísticas. Estos sistemas se usan para grandes estudios socioeconómicos, como el estudio del censo, la producción nacional o los patrones de consumo. Permiten analizar los datos desde distintos puntos de vista y mostrar todo tipo de medidas estadísticas. En la tabla siguiente, podéis ver un ejemplo de lo que se podría obtener con una de estas herramientas.

Figura 2

Contador de artículos vendidos		Barcelona	Tarragona	Lérida	Gerona
5-1-2000	Bolígrafos	15	3	7	1
	Gomas	3	1	0	5
	Portaminas	4	0	6	2

Datos estadísticos

Para acercarnos al concepto con más propiedad, en lugar de bases de datos estadísticas tendríamos que hablar simplemente de herramientas que facilitan las consultas estadísticas a bases de datos.

Lectura complementaria

Podéis ver ejemplos de tablas estadísticas en el *Anuario estadístico de Cataluña 1992-2001*. Colección Estadística de Síntesis. Departamento de Economía y Finanzas. Instituto de Estadística de Cataluña.

Suma de ingresos por artículo		Barcelona	Tarragona	Lérida	Gerona
5-1-2000	Bolígrafos	39,5	5,1	15,9	1,2
	Gomas	1,4	0,3	0	1,4
	Portaminas	8,7	0	11	5,1

Atributos acumulados

Los atributos acumulados se obtienen como resultado de aplicar una función de agregación a atributos detallados.

Hay que distinguir claramente las tablas estadísticas de las tablas relacionales. En las primeras, podemos cambiar las filas por columnas y viceversa, mientras que en las segundas no podemos. Es lo mismo tener una tabla estadística de artículos por población y una de población por artículos. Sin embargo, cada valor numérico que hay en la tabla estadística estaría en una fila diferente de una tabla relacional. Además, estos valores corresponden a atributos acumulados.

A las bases de datos estadísticas, se les pide que garanticen la confidencialidad de los datos de los individuos. Por consiguiente, su punto fuerte es la seguridad (de manera más concreta, los mecanismos de protección de inferencia de datos) y no tanto la facilidad de consulta o la presentación de los resultados, como sería de esperar si los quisieramos utilizar para tomar decisiones en una empresa. Suele tratarse simplemente de sistemas relacionales con un software añadido para mejorar la seguridad y las consultas.

Un sistema relacional ya ofrece una cierta flexibilidad en la estructuración y consulta de los datos. Sin embargo, obtener la suma acumulada de ventas combinando totales y subtotales, o determinar un cierto ranquin (dar los diez países con un mayor total de ventas), es muy difícil, si no imposible, con SQL. En el mejor de los casos, sería necesaria la intervención de un informático para hacer la consulta. El usuario no la podría hacer. Además, el modelo relacional no considera las **jerarquías de agregación** (las ciudades se agregan en regiones, las regiones en estados, etc.).

Lo que necesitamos es algo más flexible que una base de datos estadística. Después de pedir la suma de ventas por estados, nos tendría que permitir aislar uno (por ejemplo, el que tiene la mayor proporción de ventas en relación con su población) y ver las ventas desglosadas por regiones. El usuario final, sin ningún conocimiento específico de informática, debería poder navegar fácilmente por los datos.

Una base de datos relacional (estadística o no) no da la flexibilidad y facilidad de uso que requiere el análisis en línea.

1.2. Hojas de cálculo

Realmente, los analistas no utilizan las bases de datos estadísticas, principalmente debido al hecho de que son difíciles de usar. El problema no es que no se puedan usar para tareas de análisis, sino lo difícil y pesado que resulta hacerlo. Lo que sí utilizan en su día a día son las hojas de cálculo, sin ningún tipo de duda mucho más fáciles de usar. Cada celda contiene un dato que podemos referenciar fácilmente mediante sus coordenadas bidimensionales (filas × columnas). Podemos operar con los datos simplemente referenciándolos con una letra y un número. Si un dato cambia, no hay que modificar todas las fórmulas en las que aparece, sino solo cambiar el valor de una cierta celda. Este funcionamiento facilita lo que se denomina análisis *what-if* ('qué pasa si'). Es decir, ¿qué pasará si cambio este valor (por ejemplo, un precio de venta, el número de unidades producidas, el tiempo de producción, etc.)? ¿Cómo variará mi negocio?

Mecanismos de protección de inferencia

Entendemos por mecanismos de protección de inferencia todo aquello que ayude a evitar que a partir de los datos de un cierto conjunto de individuos, sea posible inferir o deducir los datos correspondientes a estos; por ejemplo, conseguir la edad de Jorge a partir de la media de edad de los habitantes de Lérida.

Navegar

En este contexto, el término *navegar* significa hacer un conjunto de consultas de modo que cuando se vea el resultado de una, se decide cuál será la siguiente.

Celdas tridimensionales

Si además de filas y columnas la hoja de cálculo permite trabajar con páginas, podemos hablar de un espacio de celdas tridimensional.

Pese a esta facilidad de uso, las hojas de cálculo todavía tienen algunas carencias:

- No son apropiadas para grandes cantidades de datos.
- No aportan ningún significado a los datos (las celdas se identifican simplemente por sus coordenadas).
- La creación de informes no es lo bastante sencilla.
- Del mismo modo que las bases de datos estadísticas, no facilitan el uso de jerarquías de agregación.

Además de esto, la posición de los datos puede determinar las operaciones que se puedan hacer. Si no se quiere tener que explicitar una por una toda la lista de celdas que intervienen en una operación, estas han de ser consecutivas, de modo que se pueda dar un rango (por ejemplo, `SUMA (D7 : D123)`). Por lo tanto, tenemos que conocer *a priori* con qué datos operaremos, para poder colocarlos en las celdas adecuadas. La pregunta que surge en este punto es si siempre habrá alguna manera de colocar los datos, de modo que podamos hacer fácilmente todas las operaciones que queremos. En los casos en que esto no sea posible, se debería tener una hoja de cálculo distinta para cada operación o conjunto de operaciones que se pueda definir dada una posición de los datos.

A pesar de que podría solucionar el problema anterior, tener muchas hojas de cálculo con los mismos datos generaría mucha redundancia, con todos los inconvenientes que, como ya sabéis, esto supone. Lo que nos hace falta es, sin perder la facilidad de uso, poder consultar y reestructurar los datos de manera fácil y flexible.

Necesitamos un sistema híbrido que nos proporcione la flexibilidad y potencia de una hoja de cálculo, y la estructura y facilidad de consulta de una base de datos.

1.3. Herramientas OLAP y multidimensionalidad

La solución actual a las necesidades de análisis de las empresas son las herramientas OLAP. Este término fue introducido por E. F. Codd en 1993, aunque ya había herramientas informáticas específicas para el análisis mucho antes.

⁽¹⁾Del inglés *fast analysis of shared multidimensional information*.

Literalmente, se refiere a la posibilidad de procesar consultas en línea (en contraposición con el procesamiento por lotes –*batch*–), con el objetivo de analizar datos. Para entender mejor lo que son estas herramientas, podemos emplear el denominado test FASMI¹. Según esta definición, un sistema OLAP ha de hacer lo siguiente:

1) **FAST**: responder la mayoría de las consultas en aproximadamente cinco segundos (excepcionalmente, podría llegar a tardar veinte). Esto hace que se tengan que implementar técnicas específicas de indexación y búsqueda, con mecanismos especiales de almacenamiento.

El tiempo de respuesta ha de ser pequeño

Estudios recientes demuestran que, si un usuario tarda más de treinta segundos en obtener el resultado de su petición, tiende a pensar que el proceso falla, a menos que se le avise de la duración. En cualquier caso, si la respuesta del ordenador tarda demasiado en llegar, los usuarios se distraen y pierden el hilo de sus razonamientos.

2) **ANALYSIS**: ofrecer herramientas de análisis estadístico y generación de informes sin que haya que programar nada (ni siquiera mediante un lenguaje de cuarta generación –4GL– como el SQL). Una herramienta OLAP podría ayudar al estudio de series temporales, la adjudicación de costes, el cambio de monedas, la prospección de datos, la definición de ratios, etc. Solo con un almacén de datos ya podríamos contestar a las preguntas "¿qué?" y "¿quién?". Con una herramienta OLAP, además de esto, también deberíamos poder responder "¿por qué?" y "¿qué pasa sí...?". Las herramientas OLAP son un complemento imprescindible de los almacenes de datos.

3) **SHARED**: implementar los mecanismos de seguridad (control de acceso y confidencialidad de los datos) y concurrencia (aunque los analistas no quieran escribir nuevos datos, sí desearán escribir y posiblemente compartir los resultados del análisis) necesarios para compartir información.

4) **INFORMATION**: ser capaz de guardar toda la información necesaria. Este punto tiene dos vertientes. En primer lugar, el volumen de datos puede llegar a ser muy grande. Por otro lado, el sistema no se tiene que limitar a contener datos, sino que también ha de registrar cuál es su significado: los metadatos (información = datos + metadatos).

Estas cuatro características son muy importantes, pero todavía queda la quinta, la más importante de todas, la que realmente distingue a las herramientas OLAP y a la que está dedicada este módulo didáctico: la **multidimensionalidad**.

Lectura recomendada

E. F. Codd; S. B. Codd; C. T. Salley (1993). *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. Arbor Software, Technical Report.

Toda herramienta OLAP ha de ser multidimensional.

Como ya hemos dicho antes, los analistas no son informáticos. Por este motivo, deben ser las herramientas las que se adapten a los mismos, y no al revés. Queremos que los usuarios no dependan del departamento de informática para hacer una simple consulta. Ahora no hablamos de un entorno transaccional con consultas predefinidas que casi nunca cambian, sino de un entorno de análisis con consultas *ad hoc* que el usuario ha de formular a medida que tiene la necesidad de ver algunos datos. Para facilitarles el trabajo, una herramienta OLAP tiene que presentar los datos como los analistas están acostumbrados a verlos, es decir, en términos de **hechos y dimensiones**, en lugar de tablas, atributos y claves foráneas.

Ejemplo de hechos y dimensiones de análisis

Imaginémonos que se quiere analizar la distribución y los valores de las ventas de una cadena de supermercados. Ventas sería nuestro hecho objeto de análisis. Un posible espacio tetradimensional para analizar este hecho estaría definido en este caso por las dimensiones **Producto**, **Cliente**, **Tiempo** y **Población** en la que se ha producido la venta.

La multidimensionalidad se basa en la dicotomía entre datos métricos (qué queremos analizar) y datos descriptivos (qué, a quiénes, dónde, cuándo, cómo, etc.). Las dimensiones (datos descriptivos) definen un espacio n -dimensional, conocido como **cubo**, en el que colocamos los hechos (datos métricos) que queremos analizar (en cierto modo, esto generaliza las hojas de cálculo, de manera que podemos tener cualquier número de dimensiones). A cada posición en este espacio la denominaremos **celda**. Cada celda corresponde a un hecho en concreto que queda determinado por las dimensiones de análisis que utilizamos. Observad la diferencia con las hojas de cálculo: las celdas no se identifican por simples caracteres mudos, sino por los valores de las dimensiones de análisis, que sí tienen asociado un significado.

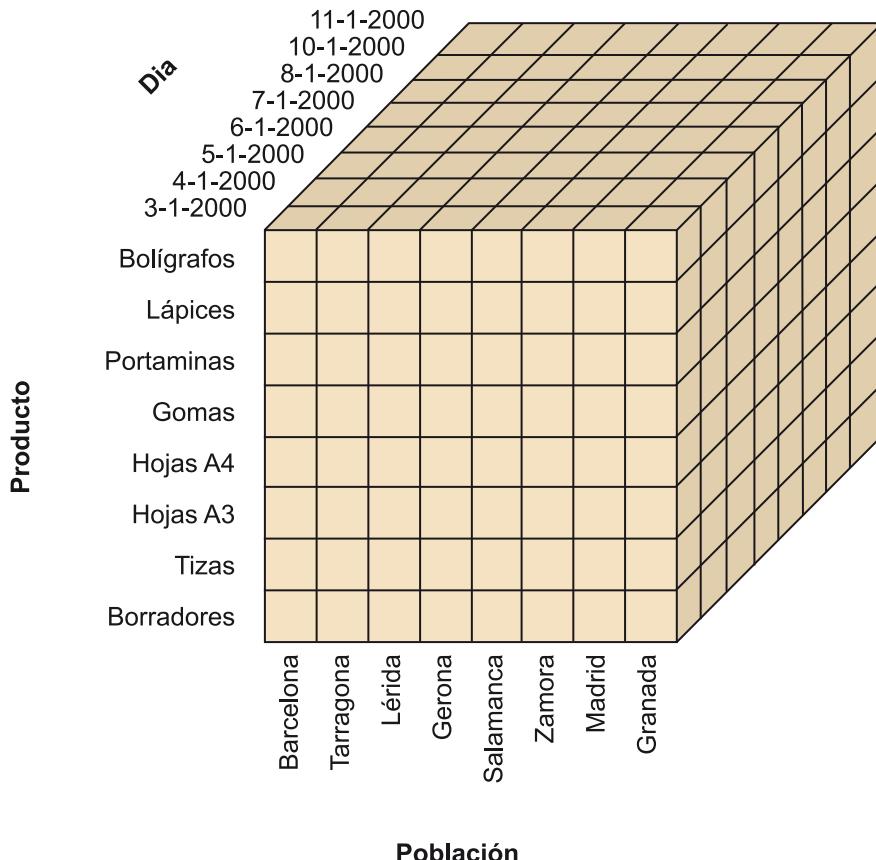
La multidimensionalidad

La multidimensionalidad no tiene sus orígenes en las bases de datos, sino que se basa en el álgebra de matrices, que ha sido utilizada para el análisis manual de datos desde el siglo XIX.

Hipercubo

Denominarlo cubo es un abuso de lenguaje, puesto que las dimensiones no necesariamente tendrán la misma longitud. Además, generalmente tendrá más de tres dimensiones. Por lo tanto, deberíamos hablar en todo caso de hipercubo.

Figura 3



Cubo tridimensional

Si para simplificar el dibujo olvidamos de manera momentánea la dimensión *Cliente*, en el caso anterior tendríamos un cubo como el de la figura 3. Cada una de las celdas de este cubo representa ventas, que es el tipo de hecho que queríamos analizar. Concretamente, la celda que hay en la intersección entre "Lérida", "Gomas" y "7-1-2000" contendrá todos los datos de los que dispongamos sobre las ventas de este artículo, en esta población y en la fecha dada.

La multidimensionalidad consiste simplemente en concebir los datos que queremos analizar en términos de hechos y dimensiones de análisis, de modo que los podemos situar en un espacio *n*-dimensional.

El cubo

El cubo no es más que una metáfora de cómo se tienen que entender los datos. No significa que las herramientas tengan que dibujar cubos *n*-dimensionales en la pantalla, ni que forzosamente deban almacenar los datos en matrices de *n* dimensiones. Este concepto sirve simplemente para ayudar a los usuarios a entender lo que pueden hacer con los datos.

Disponer de una gran base de datos que alcance a toda la empresa no tiene mucho valor para los analistas, que muy probablemente se verán sobrepasados por su volumen y complejidad. Es mucho más apropiado focalizar en un único tema (hecho). La primera contribución de la multidimensionalidad es que permite dar a cada analista solo el cubo o el conjunto de cubos que correspondan al hecho o los hechos en los que esté interesado. De este modo, reducimos la complejidad del problema y simplificamos su trabajo.

Aunque, por un lado, simplificamos la visión de los datos para que los analistas los puedan entender, por el otro, tenemos que añadir las funcionalidades que piden. En este sentido, tenemos que hablar de **niveles de detalle y jerarquías de agregación**. Los puntos que hay en cada dimensión se pueden agrupar por

APL

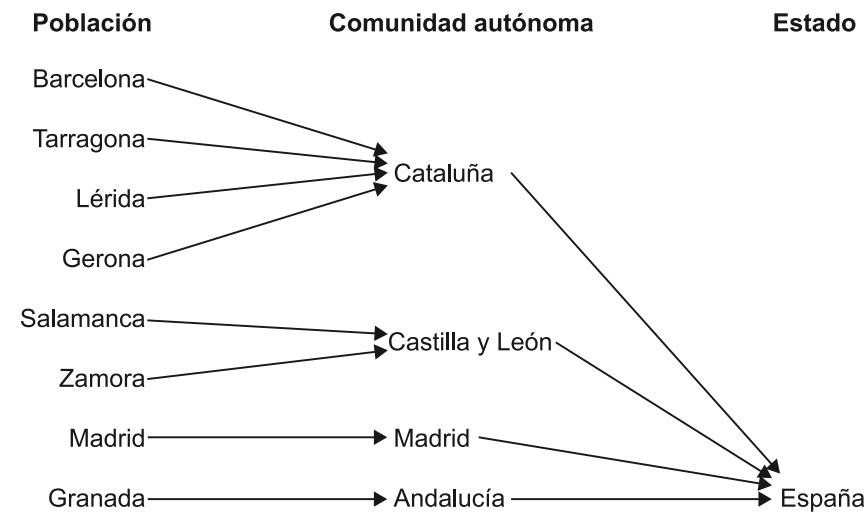
La primera herramienta multidimensional que corría sobre un ordenador fue el lenguaje de programación APL, desarrollado por IBM a finales de la década de los sesenta. Ofrecía la posibilidad de definir variables multidimensionales y operarlas mediante un conjunto de operadores específicos. A pesar de no ser muy amigable, se utilizó mucho durante toda la década de los setenta.

niveles según una cierta jerarquía. Un conjunto de puntos de un cierto nivel forman otro punto en el nivel inmediatamente superior. Esto es especialmente importante porque, para tomar decisiones, lo más habitual es mirar los datos resumidos o agregados por grupos (por ejemplo, gamas de productos, regiones geográficas, etc.).

Jerarquía de agregación de la dimensión geográfica

Podemos ver un ejemplo muy claro de jerarquía de agregación de la dimensión geográfica dibujada en la figura 4. Un conjunto de ciudades forman parte de una comunidad autónoma y un conjunto de comunidades forman un estado. De este modo, nuestra jerarquía de agregación de la dimensión geográfica tiene tres niveles de agregación diferentes: Población, Comunidad autónoma y Estado.

Figura 4

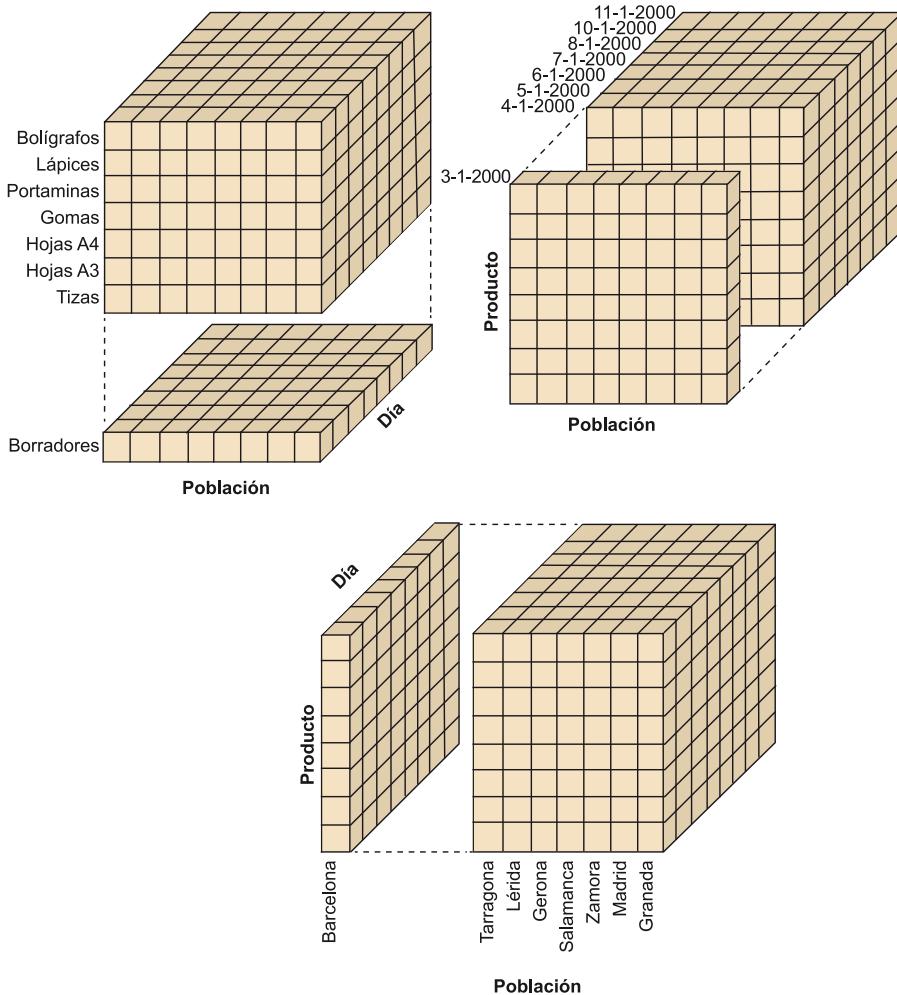


La multidimensionalidad proporciona mucho más que la simple posibilidad de visualizar los datos en forma de cubo. También nos proporciona los fundamentos para poder manipularlo de manera fácil y flexible, sin perder capacidad de cálculo. El cambio del nivel de detalle, con la posibilidad de seleccionar los elementos de las dimensiones y cambiar el objeto de análisis es lo que denominaremos navegabilidad. Podemos continuar viendo esta navegabilidad en términos de cubos. De este modo, las herramientas OLAP ofrecen, con pequeñas variaciones, las operaciones que se detallan a continuación.

*Slice*²: hace un corte al cubo de modo que se reduce el número de dimensiones. Se trata simplemente de fijar un valor a una de las dimensiones del cubo, de manera que pasamos a tener un cubo con $n-1$ dimensiones en el que todas las celdas hacen referencia al valor que hemos elegido.

⁽²⁾ Slice significa literalmente 'rodaja'.

Figura 5



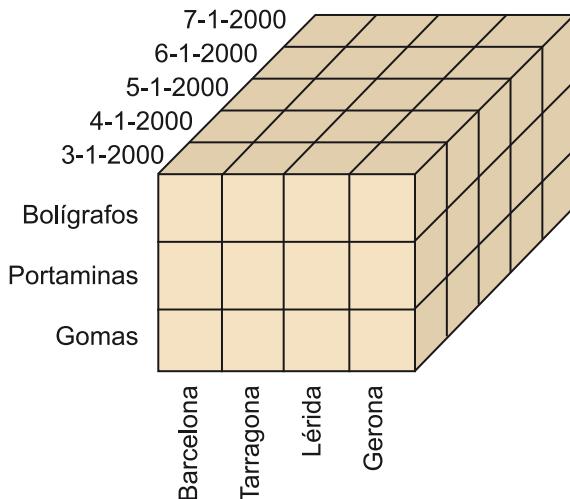
Ejemplo de slice

La figura 5 ejemplifica las tres posibles maneras de hacer *slices* en un cubo tridimensional. Fijémonos en la de arriba a la izquierda. Lo que hacemos en este caso es, de todo el cubo tridimensional que teníamos, quedarnos solo con un cubo bidimensional (una *slice*), en el que todas las celdas contienen datos referentes a "Borradores". Si a este cubo bidimensional resultante le volviéramos a hacer una *slice*, ahora para el valor "Salamanca" en la dimensión geográfica, nos quedaría un cubo unidimensional (una línea de celdas), que mostraría las ventas de borradores que ha habido en Salamanca en cada uno de los días.

*Dice*³: selecciona un subespacio del cubo original, sin reducir el número de dimensiones. Esto se consigue seleccionando un subconjunto de valores en cada una de las dimensiones.

⁽³⁾Una traducción literal al español es 'dado'.

Figura 6



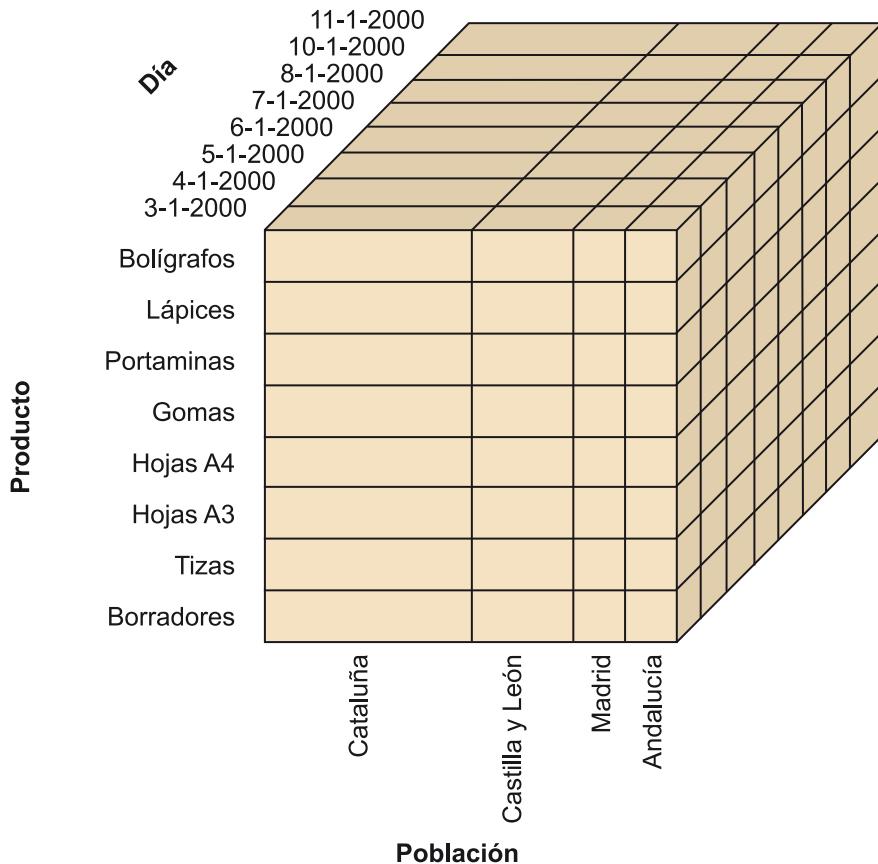
Ejemplo de *Dice*

En la figura 6, podemos ver el resultado de aplicar una operación de *Dice* al cubo tridimensional que podemos ver en la figura 3. De todo el espacio de tamaño $8 \times 8 \times 8$ que teníamos de manera originaria, nos quedamos solo con un subespacio de $4 \times 3 \times 5$, que tiene muchas menos celdas, pero que continúa siendo tridimensional.

*Roll-up*⁴: reduce el detalle con el que vemos los datos. Agrupa las celdas del cubo siguiendo una cierta jerarquía de agregación. Para obtener los datos de cada celda, aplica una cierta operación matemática (como por ejemplo, la suma o la media) a los datos de las celdas que forman parte de cada uno de los grupos.

(⁴)'Enrollar', en español.

Figura 7



Ejemplo de roll-up

La figura 7 muestra el resultado de hacer un *roll-up* hasta el nivel *Region* en el cubo que tenemos en la figura 3. Ahora, en lugar de disponer de una celda para cada una de las ciudades catalanas, tenemos una sola celda que contiene los datos que hacen referencia a todo el grupo de ciudades de Cataluña. Lo que hemos hecho es movernos desde el nivel *Poblacion* hasta el nivel *Region* dentro de la jerarquía de agregación de la dimensión geográfica que hemos explicado en el ejemplo correspondiente. Pasa lo mismo para las ciudades de Castilla y León, Andalucía y Madrid. Estos dos últimos casos son muy sencillos, porque cada región contiene solo una ciudad (en nuestro caso) y, por consiguiente, los datos de la región son los mismos que los de las ciudades correspondientes. Sin embargo, ¿qué pasa con "Cataluña" y "Castilla y León"? ¿Cómo obtenemos sus datos? Tenemos que aplicar una cierta función de agregación. Si hablamos de cantidades vendidas, la de toda Cataluña es la suma de las cantidades vendidas en Barcelona, Tarragona, Lérida y Gerona (considerando que solo tenemos tiendas en estas ciudades, o que solo nos interesan estas).

*Drill-down*⁵: aumenta el detalle con el que vemos los datos. Es la operación inversa al *roll-up*. En vez de subir en una jerarquía de agregación, bajamos y deshacemos los grupos. Observad que las funciones de agregación que hemos utilizado al hacer *roll-up* no se pueden deshacer si no disponemos de los datos originales. Tenéis que entender esta operación como un deshacer (*undo*) de la operación *roll-up*.

⁽⁵⁾Acceso al detalle subyacente de los datos.

Ejemplo de drill-down

Esta operación correspondería al paso del cubo de la figura 7 al de la figura 3. Observad que para poderlo hacer, tenemos que disponer de los datos originales. ¿Cómo conseguiríamos las ventas en cada una de las poblaciones catalanas si solo tuviéramos la suma de cantidades vendidas en toda Cataluña?

*Drill-across*⁶: cambia el tema de análisis. Después de aplicar esta operación, continuamos disponiendo del mismo espacio n -dimensional que teníamos, pero ahora las celdas contendrán datos que corresponden a un tipo de hecho diferente. En términos de álgebra relacional, el *drill-across* se asemeja a una combinación (*join*), en el sentido de que asocia cada elemento de un cubo con un elemento de otro, del mismo modo que hace la combinación entre tablas.

(⁶)*Drill-across* literalmente significa 'agujerear a través'. Realmente, se utiliza por similitud con *drill-down*.

Ejemplo de *drill-across*

Con esta operación, si partimos del cubo de la figura 6, obtendremos un cubo igual que aquel pero que, en lugar de contener datos de ventas, contiene datos de producción, por ejemplo. La celda que hay en la intersección entre "Lérida", "Gomas" y "7-1-2000" contendrá todos los datos que tengamos sobre la producción de este artículo, en esta población y en la fecha dada.

Las dimensiones se utilizan para seleccionar y agregar los datos al nivel de detalle deseado.

Estas operaciones, junto con el cambio de orden de los elementos que forman las dimensiones y el cambio de posiciones de estas (dicho de otro modo, hacer una rotación o pivotar), son mejoras importantes respecto a la rigidez en las filas y columnas de una hoja de cálculo. Sin embargo, dejando de lado la navegabilidad y esta flexibilidad de presentación, las herramientas OLAP también tienen que ofrecer facilidades para hacer informes. Aunque en pantalla fueran capaces de representar tres o más dimensiones, en papel esto resulta claramente complicado. Podéis encontrar representaciones gráficas más o menos sofisticadas, pero podríamos decir que las dos posibilidades de representación más habituales son las tablas estadísticas (como, por ejemplo, las de la figura 2) y las tablas relacionales (por ejemplo, las de las figuras 8 y 9).

Figura 8

Día	Producto	Población	Número de artículos	Ingresos
5-1-2000	Bolígrafos	Barcelona	15	39,5
5-1-2000	Gomas	Barcelona	3	1,4
5-1-2000	Portaminas	Barcelona	4	8,7
5-1-2000	Bolígrafos	Tarragona	3	5,1
5-1-2000	Gomas	Tarragona	1	0,3
5-1-2000	Portaminas	Tarragona	0	0
5-1-2000	Bolígrafos	Lérida	7	15,9
5-1-2000	Gomas	Lérida	0	0
5-1-2000	Portaminas	Lérida	6	11,0
5-1-2000	Bolígrafos	Gerona	1	1,2
5-1-2000	Gomas	Gerona	5	1,4

Día	Producto	Población	Número de artículos	Ingresos
5-1-2000	Portaminas	Gerona	2	5,1

Figura 9

Día	Producto	Población	Número de artículos	Ingresos
5-1-2000	Bolígrafos	Cataluña	26	61,7
5-1-2000	Gomas	Cataluña	9	3,1
5-1-2000	Portaminas	Cataluña	12	24,8

Ejemplo de informe

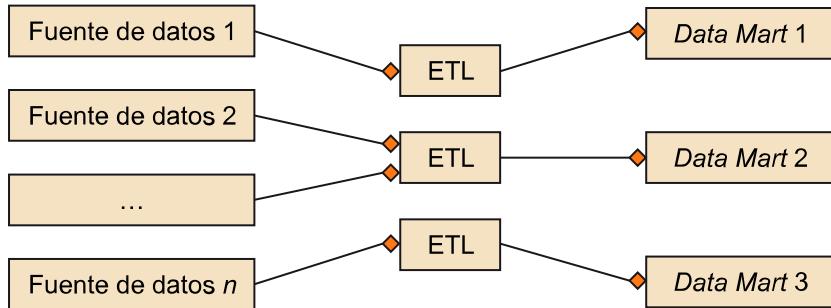
Pensemos que, del cubo de la figura 6, solo nos quedamos con la rodaja que contiene los datos del día "5-1-2000". Además, imaginémonos que cada celda contiene tanto el número de unidades vendidas como la cantidad de euros ingresada. En la tabla relacional de la figura 8, podéis ver el informe que resultaría de esta consulta. Si ahora hiciéramos un *roll-up* hasta el nivel *Region*, obtendríamos el informe de la tabla relacional de la figura 9, en la que tenemos resumidos los datos para toda Cataluña (hemos sumado los datos de las ciudades).

La simplicidad de la concepción multidimensional de los datos supone dos beneficios. Por un lado, ayuda a los analistas a entender los datos. Por el otro, ayuda a los informáticos a prever las consultas que harán los analistas, de modo que puede optimizar el tiempo de respuesta con mucha más facilidad.

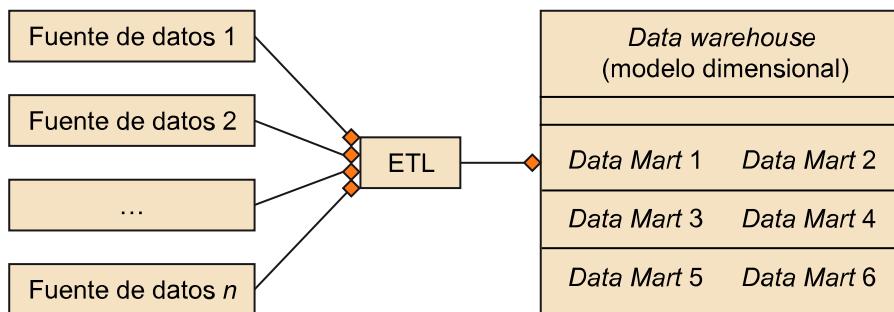
1.4. Modelización de un almacén de datos y multidimensionalidad

El concepto de almacén de datos llegó de la mano de Bill Inmon y Ralph Kimball. Ambos pensaron en un único repositorio de información para poder integrar y explotar información de diversos sistemas fuentes. Pero, más allá de esta generalización conceptual, cada uno propuso su propio enfoque teórico:

- a) Kimball sugiere utilizar una metodología *Bottom-Up*. Tal como puede verse en la figura 10, la información se extrae de los sistemas transaccionales para ser cargada en diferentes *Data Mart*. Dichos *Data Mart* han sido creados independientemente, modelados de acuerdo con un modelo de datos multidimensional y tienen foco departamental. Generalmente son implementados con tecnología ROLAP o MOLAP.

Figura 10. Visión *Bottom-up*

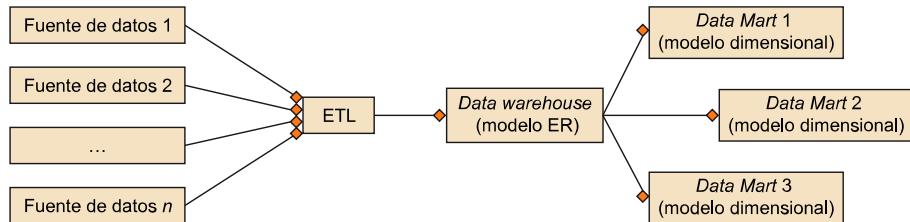
Este modelo debe evolucionar a lo largo del tiempo para formar un *Data Warehouse* (DW) tal y como se observa en la figura 11. Los *Data Mart*, en este caso, estarían todos en un mismo repositorio, también respetando el modelo dimensional y se relacionarían entre sí mediante sus dimensiones (dimensiones conformadas).

Figura 11. Evolución de la visión *Bottom-up*

La arquitectura en BUS de Kimball expresa que la información no procesada es transformada a un formato presentable en lo que él concibe como el *staging area*, siempre consciente de la productividad y la calidad. Todo comienza con extracciones coordinadas de los sistemas fuente.

b) Inmon nos ofrece una visión *Top-Down*. Coincide con el segundo caso que vimos de Kimball donde el DW se nutre de una sola ETL, pero en este caso el DW no está modelado dimensionalmente, sino que está en tercera forma normal (3NF). Tal como puede verse en la figura 12, Inmon, sitúa el almacén de datos en el centro de la FIC (*Corporate Information Factory*) proporcionando un marco lógico para la entrega de inteligencia de negocio. Entiende que esta forma es mucho más rica y adaptable que el modelo de Kimball.

Una vez que tenemos el *Data Warehouse* generado de esta manera, se pueden crear los *Data Mart* para las áreas de negocio que necesitemos, y además lo podríamos utilizar para cualquier otro tipo de sistema decisional como, por ejemplo, sistemas expertos, o minería de datos.

Figura 12. Esquema *Top-down*

Inmon cree que su aproximación, al utilizar *Data Mart* dependientes como la fuente de un esquema tipo estrella, permite resolver el problema de acceso de toda la empresa a la misma información, la cual puede cambiar con el tiempo.

Pero el problema es que es más costoso de mantener y de implementar. El de Inmon es un modelo que mira a largo plazo y, para una metodología ágil, el largo plazo es secundario. Para adaptarlo y no perder la agilidad de, por ejemplo, el primer modelo de Kimball, se permiten grupos de datos repetidos, lo que viola las reglas de normalización, pero consigue un alto nivel de desempeño, que permite un fácil acceso al usuario final con una adecuada velocidad de respuesta.

No hay una opción correcta o incorrecta entre estas dos visiones, ya que representan diferentes filosofías de almacenamiento de datos.

En realidad, los almacenes de datos en la mayoría de las empresas están más cerca de la idea de Ralph Kimball. Esto se debe a que la mayoría de los almacenes de datos surgen en el ámbito de un departamento, y por lo tanto se originan como un *Data Mart*. Posteriormente, cuando ya se han incorporado nuevos *Data Mart*, se consigue que el conjunto de todos ellos se convierta en un auténtico almacén de datos.

En consecuencia, los cubos OLAP usados en las organizaciones suelen corresponderse con los distintos *Data Mart* existentes, ya que aprovechan de forma directa su diseño multidimensional.

Pese a las diferencias que se pueden apreciar de manera inmediata entre ambas arquitecturas, también existen elementos en común. Todas las empresas requieren almacenar recursos, analizar e interpretar la información que generan y acumulan a lo largo del tiempo con el fin de tomar las mejores decisiones, aquellas que maximicen su valor. Por ello resulta prioritario crear sistemas de análisis y retroalimentación que permitan comprender la información propia (que reside principalmente en el *Data Warehouse*) y, de esta manera, contar con los elementos adecuados para una sólida toma de decisiones.

Inmon y Kimball coinciden en que los *Data Mart* (o *Independent Data Warehouse*) no satisfacen las necesidades de precisión y oportunidad de la información, ni facilitan el acceso para los usuarios. Estos sistemas se construyen para satisfacer necesidades específicas, sin ver los otros procesos de análisis de la infor-

mación. Las extracciones múltiples y descoordinadas de las mismas fuentes de datos son ineficientes y solo provocan el desperdicio de los recursos, pues generan reglas y convenciones de negocio similares pero con variaciones e inconsistencias en los nombres, lo que origina confusión dado que en ocasiones se realizan las mismas operaciones más de una vez. Lo que se tiene al final, en la toma de decisiones basada en datos independientes, es una atmósfera de incertidumbre y duda.

Finalmente, en lo relativo al desarrollo del almacén de datos, el modelo de Inmon ve al departamento de IT como el proveedor y desarrollador del *Data Warehouse*, mientras que Kimball ve a un equipo conformado igualmente por personas de IT y parte de los usuarios finales.

El modelo de Inmon sería recomendable para empresas que tienen un gran equipo de especialistas en *Data Warehouse*, que van a desarrollar un proyecto amplio para toda la empresa, que va a almacenar datos que no son exclusivamente métricas de negocios y que puede esperar un plazo largo para ver los resultados (de cuatro a nueve meses).

El modelo de Kimball es adecuado para proyectos que se van a realizar por partes (*Data Mart*), estimando la primera en unos noventa días para luego dedicar de sesenta a noventa días en cada una de las subsiguientes. Conviene abordarlos desde una visión completa de FIC: dotando a cada *Data Mart* de los elementos necesarios para asegurar la integridad entre ellos; evitar replicar ETL o movimientos de datos ya existentes en otros *Data Mart*; crear dimensiones y métricas conformadas; compartir áreas de *staging*, etc.

2. Componentes del modelo multidimensional

Ahora que ya habéis visto qué es una herramienta OLAP y los conceptos básicos de la multidimensionalidad, lo formalizaremos estudiando por separado las estructuras, operaciones y restricciones de integridad propias de un modelo multidimensional. Este modelo es independiente de cualquier herramienta y os servirá para aclarar los conceptos generales. Podéis considerar que lo que veréis en este apartado supone para las herramientas OLAP lo mismo que el modelo relacional supone para las bases de datos relacionales.

Los tres componentes de un modelo de datos

Todo modelo de datos está formado por tres componentes: estructuras de datos, operaciones sobre los datos y restricciones de integridad inherentes al mismo modelo. En el caso del modelo relacional, las estructuras son las relaciones; los conjuntos de operaciones, por ejemplo, el álgebra relacional o el lenguaje SQL; y las restricciones de integridad, la unicidad y entidad de la clave primaria, la integridad referencial y la integridad de dominios.

2.1. Estructuras de datos

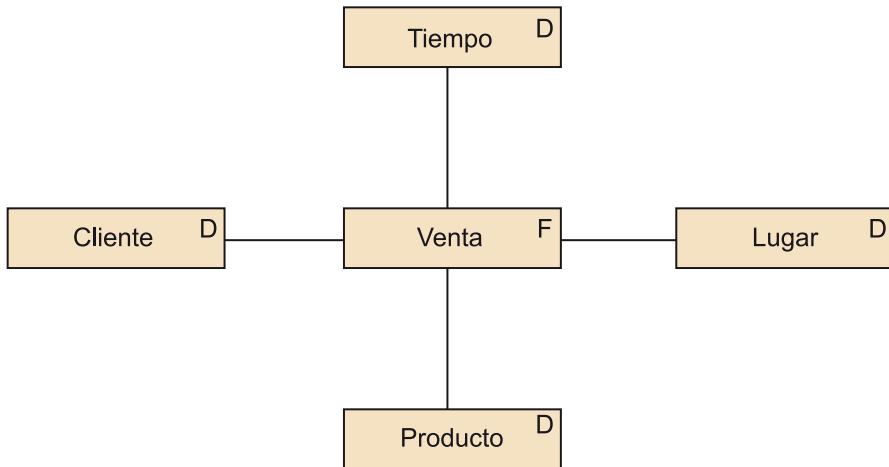
El modelo multidimensional está marcado por la dicotomía hecho-dimensión. Todos los elementos deben estar a un lado o al otro de esta línea divisoria. Por consiguiente, lo primero que hay son Hechos y Dimensiones.

Un Hecho representa un tema objeto de análisis.

Una Dimensión representa un punto de vista que utilizaremos en el análisis de los datos.

Para dibujar los elementos y las relaciones entre estos, utilizaremos la notación de UML. Dimensiones y Hechos son clasificadores. Por lo tanto, los dibujaremos con rectángulos con el nombre en su interior, pero pondremos una *D* o una *H* en la esquina superior derecha para distinguir unos de otros. Relacionaremos cada Hecho con sus Dimensiones mediante asociaciones.

Figura 13



Ejemplo de Hechos y Dimensiones

En la figura 13, podéis ver un esquema con un Hecho y cuatro Dimensiones de análisis. Queremos analizar las ventas según cuándo y dónde se produjeron, y qué y a quiénes se vendió.

Los clasificadores (Hechos y Dimensiones) contienen otros elementos que dan más detalle sobre los datos. Empezamos primero por analizar el contenido de las Dimensiones.

2.1.1. Dimensiones

Sabemos que una dimensión representa un punto de vista desde el cual se pueden analizar los datos. Consideraremos una dimensión específica y a partir de aquí definiremos los conceptos.

Dentro de una dimensión, podemos distinguir grupos de instancias según su tamaño (granularidad).

Un Nivel representa un conjunto de instancias de una Dimensión que tienen la misma granularidad, y lo dibujaremos como una clase (un rectángulo con los nombres dentro) con una *N* en la esquina superior derecha.

Dimensión

Dimensión viene del latín *dimetiri*, que se traduce por 'medir' (por ejemplo, decimos que las dimensiones de una nevera son $60 \times 60 \times 180$). Realmente, los romanos tomaron prestada la palabra de los griegos, que la utilizaban en un sentido mucho más filosófico, equivalente a la acepción que encontramos actualmente en la geometría (por ejemplo, hablamos de espacio tridimensional).

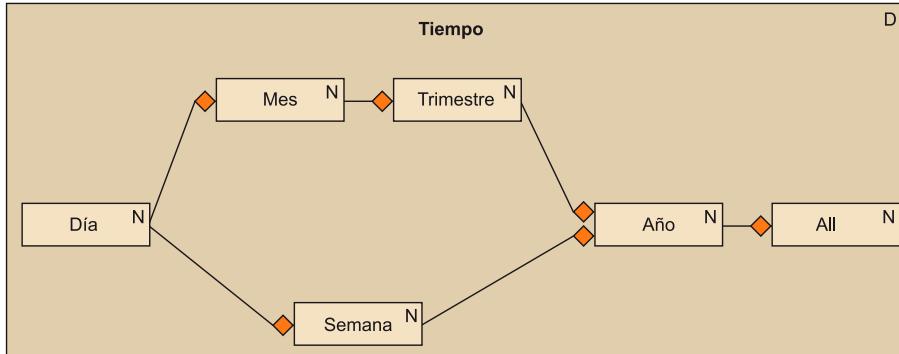
Ejemplo de granularidades

Dentro de la Dimensión Lugar, tendríamos que las poblaciones poseen una granularidad más pequeña que las regiones y estas tienen una granularidad más pequeña que los estados. Por lo tanto, representaremos con una misma clase (Nivel) todas las poblaciones, pero tendremos una clase diferente para las regiones y otra más para los estados. Las tres clases representan lugares y, por lo tanto, estarán dentro de la misma Dimensión.

Las instancias de un cierto Nivel se agrupan para dar lugar a instancias de otro Nivel de granularidad más grande. Podemos decir que las instancias de un Nivel forman instancias de otro Nivel, o que hay relaciones parte-todo entre

Niveles. Representaremos estas relaciones con agregaciones entre los Niveles y distinguiremos un Nivel especial, que denominaremos `All`, y que representa la agrupación de todas las instancias de la Dimensión al mismo tiempo.

Figura 14



Ejemplo de jerarquía de agregación no lineal

En la Dimensión temporal dibujada en la figura 14, podemos ver que un conjunto de días da lugar a un mes, pero un conjunto de días también puede dar lugar a una semana. También podemos agrupar meses para obtener trimestres y podemos obtener años por agrupación de semanas o trimestres. Finalmente, vemos un Nivel especial `All` que representa el grupo formado por todos los años que estamos interesados en analizar.

Generalmente, los Niveles dentro de una Dimensión y las agregaciones que los unen forman un grafo dirigido, conocido como jerarquía de agregación.

De los axiomas de la mereología se pueden deducir las siguientes propiedades de este grafo.

- No puede contener ciclos.
- Contiene un único Nivel que no tiene partes (lo denominan atómico).
- Puede haber o no un Nivel `All`, pero si está:
 - Solo hay uno.
 - Contiene exactamente una instancia.
 - No es parte de ningún otro Nivel.
- Todos los Niveles que no son parte de ningún otro Nivel se pueden conectar directamente al Nivel `All`.
- Todas las instancias (menos las del Nivel atómico) han de tener al menos una parte.
- Todas las instancias (menos las del Nivel atómico) pueden tener más de una parte.

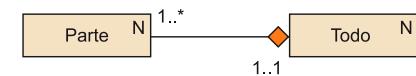
Mereología

La mereología es la ciencia que estudia las relaciones parte-a-partes.

- Los conjuntos de partes de dos instancias de un mismo nivel no tienen que ser necesariamente disyuntos.
- Siempre podemos construir el grafo de manera que todas las instancias participen en un todo, menos la del Nivel All.

Figura 15

Opción b)

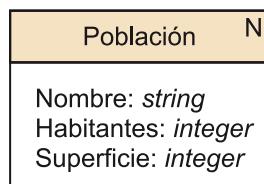


Opción a)



De estas propiedades, deducimos que las posibles multiplicidades de las agrupaciones que hay dentro de una Dimensión son solo las que tenéis dibujadas en la figura 15 (en algunos casos particulares, podríamos encontrar que en lugar del asterisco hay un valor concreto para los hitos superiores). La única elección que hay realmente es si una parte puede participar en un único todo (opción a) o en más de uno (opción b). Lo más habitual es la opción a; por lo tanto, si no se indica nada, pensaremos que la multiplicidad que tenemos es esta.

Figura 16



Como podéis ver en la figura, un Nivel tiene asociados un conjunto de atributos, los denominados Descriptores.

Los atributos que podemos encontrar en un Nivel contienen la información no jerárquica y están definidos sobre un dominio discreto. Se denominan Descriptores.

Los Descriptores no se utilizan para formar grupos, sino simplemente para seleccionar instancias o mostrarlos en los informes.

Relacionando los nuevos conceptos, vemos lo siguiente:

Una Dimensión contiene un conjunto de Niveles relacionados por agrupaciones. Cada uno de los Niveles tiene atributos que denominamos Descriptores.

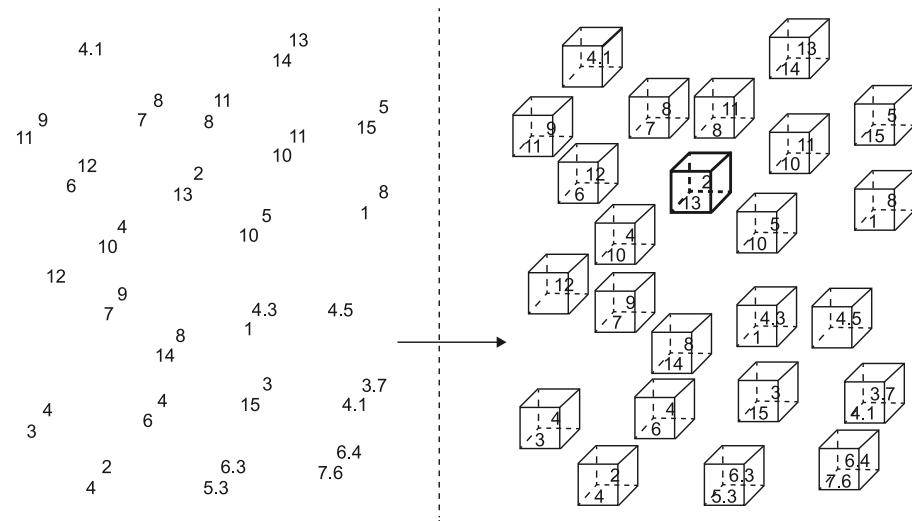
Finalmente, a pesar de que el análisis multidimensional se basa en el álgebra de matrices, hay que decir que, al contrario que en el caso de los espacios lineales, el modelo multidimensional, por sí mismo, no incluye ningún tipo de orden o distancia entre las instancias de una Dimensión. Si habláramos de la Dimensión temporal, podríamos definir fácilmente este orden (las diez de la mañana van antes que las tres de la tarde y están separadas por cinco horas). Sin embargo, ¿cómo lo haríamos con la Dimensión de productos o clientes?

2.1.2. Hechos

Olvidémonos por un momento de las Dimensiones y pensemos que queremos analizar los hechos. Tenemos un conjunto inmenso de datos (que denominaremos **mediciones**, en un sentido amplio de la palabra) dentro de nuestro almacén de datos. Lo que queremos es poner un poco de orden en este conjunto inabordable y acercar los datos a los analistas. Lo primero que hay que hacer es representar las mediciones que hacen referencia al mismo acontecimiento⁷ dentro de una misma estructura. Denominaremos a esto celda (escrito con minúsculas).

⁽⁷⁾Cualquier suceso o concepto susceptible de ser analizado.

Figura 17



Ejemplo de celdas

En la parte izquierda de la figura 17 tenemos un conjunto de mediciones: cantidades producidas, cantidades vendidas, costes, ingresos, etc. En la parte derecha de la misma figura, hemos agrupado las mediciones que hacen referencia al mismo acontecimiento. Por ejemplo, la celda en negrita podría corresponder a la venta que se hizo a Jorge hace un par de días. Le vendimos dos objetos y le cobramos trece euros.

Pese a esta primera agrupación de mediciones en celdas, todavía no cumplimos los requerimientos de los usuarios. El paso siguiente es poder unir celdas para obtener otras más grandes, que no solo representen un acontecimiento, sino muchos a la vez (por ejemplo, todas las ventas que se hicieron anteayer: la de

Jorge, la de Juan, la de Pedro, etc.). El conjunto de todas las celdas C con la unión \cup forma un semigrupo commutativo. Es decir, cumple las propiedades siguientes:

- Cerrado (la unión de dos celdas siempre es otra celda).

$$\forall x, y \in C \quad x \cup y \in C$$

- Conmutativo (no importa el orden en que hagamos una unión: obtendremos la misma celda).

$$\forall x, y \in C \quad x \cup y = y \cup x$$

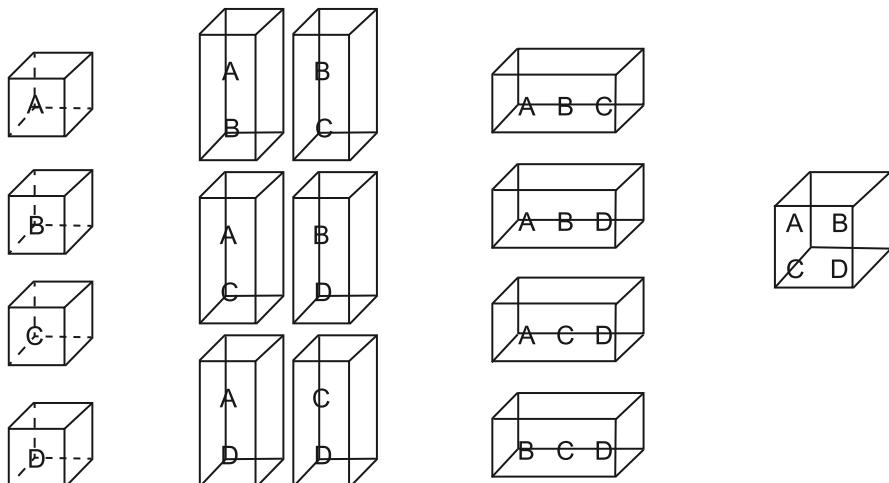
- Asociativo (podemos priorizar una secuencia de uniones como queramos, sin alterar la celda resultado).

$$\forall x, y, z \in C \quad x \cup (y \cup z) = (x \cup y) \cup z$$

- Elemento neutro (hay un elemento que, operado con cualquier otro, no lo modifica).

$$\forall x \in C \quad x \cup \emptyset = x$$

Figura 18

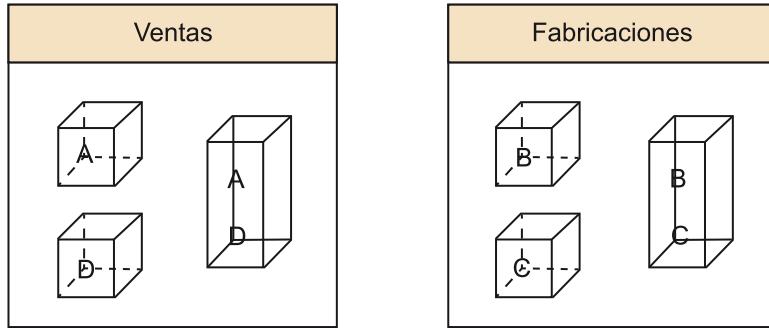


Si denominamos C_A al conjunto de todas las celdas atómicas⁸ y permitimos todas las uniones posibles, obtenemos que C contiene $2^{\text{Card}(C_A)} - 1$ celdas (que es la cardinalidad del conjunto de partes de C_A , $\text{Card}(P(C_A))$). En la figura 18, podéis ver todas las celdas que se pueden obtener a partir de cuatro celdas ató-

⁽⁸⁾Las celdas atómicas son aquellas que no podemos obtener como resultado de la unión con otras celdas.

micas. La cardinalidad de C crece de manera exponencial respecto al número de celdas atómicas. Por consiguiente, no hay que tener muchas celdas atómicas para que el problema de guardar o consultar C se haga intratable.

Figura 19



Pensemos ahora en el significado de esta unión de celdas. ¿Qué sentido tiene hacer la unión de una celda que representa ventas con una celda que representa la fabricación de un cierto producto? ¿Qué tipo de celda obtendremos como resultado? ¿Una venta? ¿Una fabricación? No tiene mucho sentido unir celdas de tipos distintos. Si restringimos la unión a celdas del mismo tipo (el mismo Hecho), ya solo tenemos $\sum_i \text{Card}(P(C_{Ai}))$ celdas, siendo C_{Ai} el conjunto de celdas atómicas instancia del Hecho i . En la figura 19, podéis ver qué celdas podemos obtener si consideramos que A y D son de un tipo de Hecho, y B y C de otro tipo.

El conjunto de celdas se ha reducido de manera drástica. Sin embargo, todavía hay muchas que no interesan a los analistas. Las celdas ganan significado solo cuando están asociadas a instancias de las Dimensiones. Únicamente cuando sabemos qué producto se vendió, a quién se lo vendimos, dónde lo vendimos, etc. la celda consigue todo su significado. Por lo tanto, solo interesan las celdas que están vinculadas a una instancia de cada una de las Dimensiones. No agrupamos cualquier conjunto de celdas, sino que utilizamos como criterio de agrupación las Dimensiones. No agruparemos una celda que contenga datos mensuales con otra que contenga datos trimestrales. ¿En qué nivel de la Dimensión temporal estaría el resultado de esta unión? Lo que sí haremos es agrupar tres celdas mensuales para obtener una celda trimestral.

Las instancias de las Dimensiones

Recordad que las instancias de las Dimensiones están clasificadas por Niveles. Por lo tanto, cada celda está asociada a un Nivel en cada una de las Dimensiones.

Al conjunto de celdas del mismo Hecho que están asociadas a instancias del mismo Nivel para cada una de las Dimensiones, lo denominaremos Celda⁹.

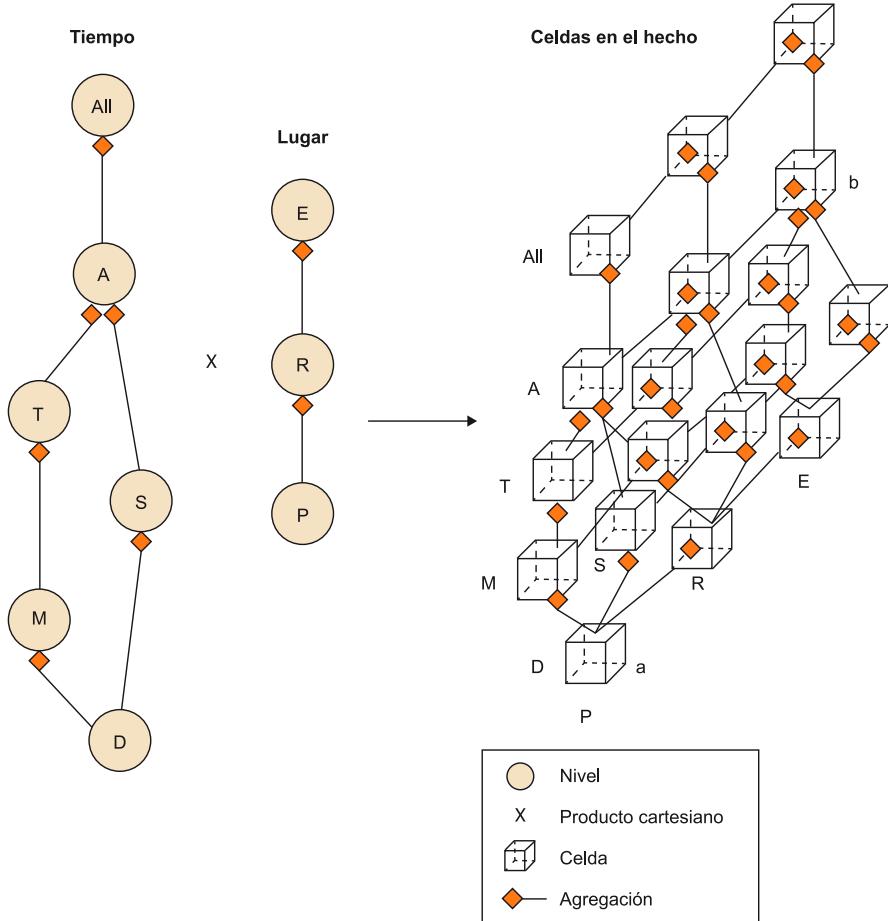
⁹⁾Utilizaremos la mayúscula para distinguir los conjuntos de celdas de las celdas individuales.

Una Celda (que dibujaremos como una clase con una C en la esquina superior derecha) representa un conjunto de instancias de un Hecho que tienen la misma granularidad.

Dentro de un Hecho, tendremos tantas Celdas como elementos hay en el producto cartesiano de los Niveles de las Dimensiones. Al igual que los Niveles, estas Celdas estarán relacionadas por agregaciones. Esto indica que las instancias de una Celda componen las instancias de la Celda inmediatamente superior.

Las instancias de las celdas se agrupan en jerarquías de agregación.

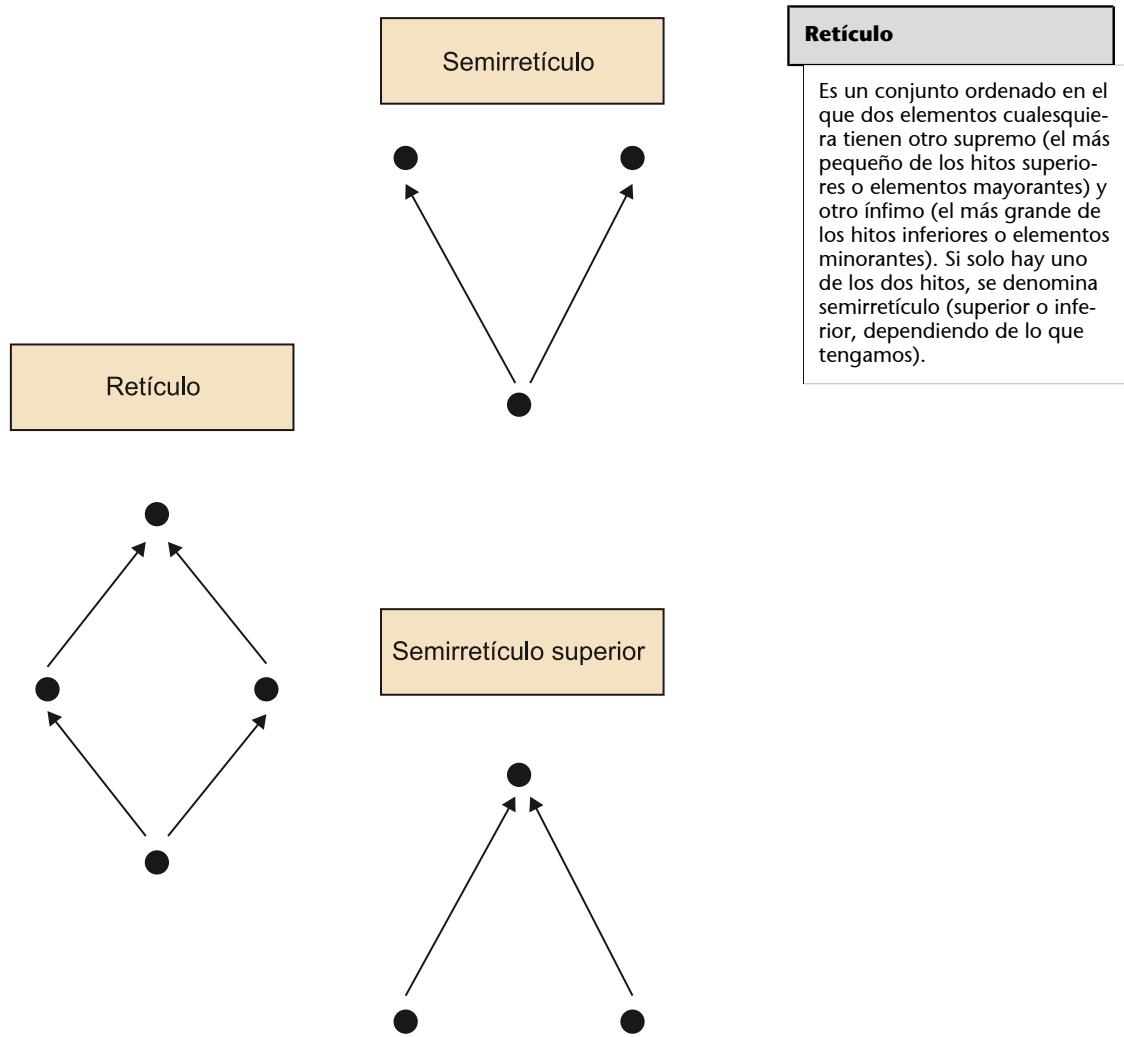
Figura 20



Ejemplo de Celdas en un Hecho

En la figura 20, tenéis dibujado un esquema con las dos Dimensiones que hemos visto hasta ahora: Tiempo y Lugar. La primera tiene seis Niveles y la segunda solo tiene tres. Para cada combinación posible de Niveles de estas Dimensiones, tenemos una Celda distinta en nuestro Hecho. Por ejemplo, la Celda "a" corresponde a las granularidades Día-Población, la Celda "b" corresponde al Nivel Año-Estado, y así hasta llegar a las dieciocho (seis por tres) combinaciones que hay. Todas las instancias de la Celda "b" están asociadas con una instancia de Año y con otra de Estado.

Con esto, obtenemos que cada Hecho contiene un grafo con estructura de semirretículo inferior, que será un retículo solo si todas las Dimensiones también lo son. Este grafo nos muestra cómo podemos agrupar las celdas para obtener otras más complejas.



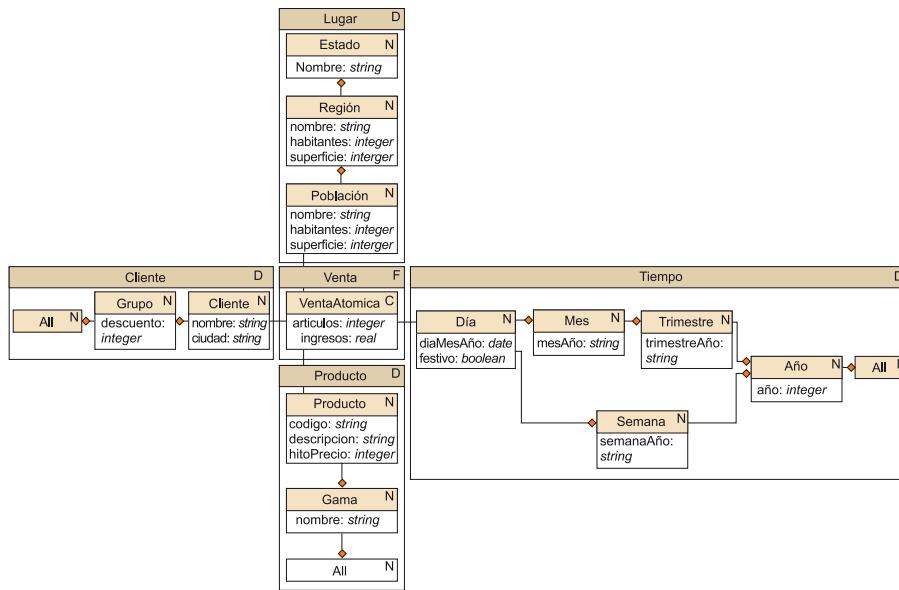
Una Medida es un atributo de una Celda.

Los valores de las Medidas (que antes hemos denominado mediciones) de una celda se obtienen como función de las mediciones de las celdas que la componen. Al contrario de lo que ocurre con los Descriptores, la mayor parte de las Medidas son de tipo numérico, a pesar de que algunas son booleanas o, incluso, textuales.

Una vez llegados a este punto, podemos relacionar los conceptos vistos en este apartado:

Un Hecho contiene un conjunto de Celdas (con C mayúscula) relacionadas por agregaciones. Cada una de las Celdas tiene atributos que denominamos Medidas.

Figura 21



Ejemplo de esquema multidimensional

La figura 21 muestra un esquema multidimensional entero. Debido a la forma que tienen, se suelen denominar esquemas en estrella. Para simplificarlo, solo se dibujan las Celdas que tienen especial interés (en este caso, solo la Celda asociada al Nivel atómico de cada Dimensión).

Como podéis ver en la figura 21, asociamos las Celdas con los Niveles correspondientes. La multiplicidad de estas asociaciones siempre es *–1. Cada celda se asocia con una sola instancia de un Nivel. Por consiguiente, no hay que explicitar esta multiplicidad.

Los principales elementos de un esquema multidimensional son, por un lado, las Dimensiones, los Niveles y los Descriptores; y por el otro, de manera simétrica, los Hechos, las Celdas y las Medidas.

2.2. Operaciones sobre los datos

En este apartado, veremos un conjunto de operaciones algebraicas sobre cubos. A pesar de que hemos dicho que la multidimensionalidad se fundamenta en la representación de los datos en forma de cubos, todavía no hemos visto qué es un cubo. Un cubo es una función inyectiva que va de un espacio n -dimensional finito (definido por el producto cartesiano de n Niveles $\{N_1, \dots, N_n\}$) al conjunto de instancias de una Celda (C_c).

$$c(x): N_1 \times \dots \times N_n \rightarrow C_c$$

Un cubo simplemente es una función que dice qué celda va en cada punto del espacio. Por lo tanto, podemos hacer con este cualquier cosa que haríamos con una función (por ejemplo, componer o unir dos de los mismos).

Figura 22

Hecho \Rightarrow Drill-across	Dimensión \Rightarrow CambioBase
Celda \Rightarrow -	Nivel \Rightarrow Roll-up
Medida \Rightarrow Proyección	Descriptor \Rightarrow Selección

Cada operación afecta directamente a un solo tipo de elemento del modelo (podéis ver las correspondencias en la tabla de la figura 22): *drill-across* permite seleccionar un Hecho; *CambioBase* permite seleccionar el conjunto de Dimensiones que utilizaremos; *roll-up* permite seleccionar el Nivel de detalle en cada Dimensión; *Proyección* permite elegir las Medidas que queremos ver, y *Selección* hace que podamos utilizar los Descriptores para elegir las instancias concretas de cada Nivel que queremos ver. ¿Por qué no hay ninguna operación asociada a Celda? ¿Cómo podemos elegir la Celda que queremos ver? No puede haber una operación asociada a Celda porque entraría en conflicto con el *roll-up*. Puesto que la Celda queda determinada por los Niveles que elegimos, no podemos elegir Niveles y Celdas al mismo tiempo.

Veamos las definiciones de estas cinco operaciones.

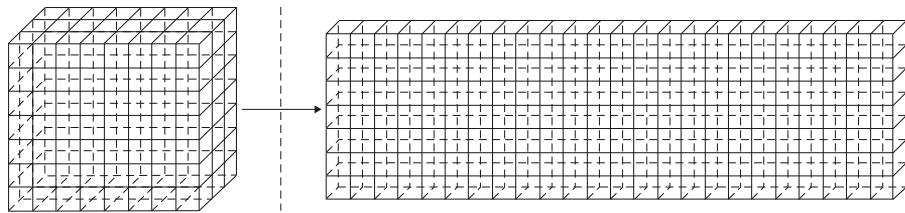
Drill-across: cambia el objeto de análisis, es decir, el Hecho. Para poder hacerlo, debe haber alguna relación entre el Hecho que tenemos y el Hecho que queremos tener. Necesitamos una función inyectiva entre los dos, que determine por qué celda del cubo destino tenemos que sustituir cada celda del cubo origen. El caso más común es que los dos Hechos comparten Dimensiones de modo que los mismos Niveles identifiquen las celdas de los dos cubos.

$$c_{destino}(x) = \text{Drill-across}_f(c_{origen}) = f(c_{origen}(x))$$

Proyección: simplemente selecciona el conjunto de Medidas que queremos ver de entre las que hay disponibles en la Celda del cubo origen. Equivale a la operación homónima del álgebra relacional.

$$c_{destino}(x) = \text{Proyección}_{m_1, \dots, m_k}(c_{origen}) = c_{origen}(x)[m_1, \dots, m_k]$$

Figura 23



CambioBase: redistribuye el mismo conjunto de celdas dentro de otro espacio. Para aplicarlo, necesitamos disponer de una función inyectiva entre los dos espacios (es decir, entre Dimensiones), de modo que cada punto del espacio del cubo destino determine un punto del cubo origen. Esto puede servir simplemente para reordenar los puntos del espacio, o también puede cambiar el número de Dimensiones (como está esquematizado en la figura 23), pero no cambia el número de celdas, ni su contenido.

$$c_{destino}(x) = \text{CambioBase}_f(c_{origen}) = c_{origen}(f(x))$$

Roll-up: agrupa las celdas de un cubo basándose en una jerarquía de agregación. Aumenta la granularidad hasta un cierto Nivel. Reduce el número de celdas, pero no el de Dimensiones.

$$c_{destino}(x) = \text{Roll-up}_{\text{Nivel}}(c_{origen}) = \bigcup_{r(y)=x} c_{origen}(y) \quad 4.1$$

Selección: mediante un predicado lógico sobre los Descriptores de Niveles, selecciona un conjunto de puntos dentro del espacio n -dimensional. Es absolutamente equivalente a la operación homónima del álgebra relacional.

$$c_{destino}(x) = \text{Selección}_{\text{predicado}}(c_{origen}) = \begin{cases} c_{origen}(x), & \text{si } \text{predicado}(x) = \text{cierto} \\ \text{indefinido}, & \text{si } \text{predicado}(x) = \text{falso} \end{cases}$$

4.2

Este conjunto de operaciones es cerrado, es decir, los operandos son cubos y el resultado también. Esto implica que podemos concatenar las operaciones. Por ejemplo, la operación *Slice* que hemos visto en el apartado "Herramientas OLAP y multidimensionalidad" sería equivalente a elegir un punto mediante la operación *Selección* y hacer un cambio de base para reducir en uno el número de Dimensiones. Observad que la Dimensión que eliminamos con el cambio de base solo contiene la instancia "k" y, por lo tanto, a pesar de que perdemos una Dimensión, no perdemos ninguna celda (el espacio $a \times b \times 1$ tiene el mismo número de puntos que el $a \times b$).

$$Slice_{Ni=k}(c_{origen}) = CambioBase_{f:N1 \times \dots \times Nn \rightarrow N1 \times \dots \times Ni-1 \times Ni+1 \times \dots \times Nn}(Selección_{Ni=k}(c_{origen}))$$

Otra operación que podéis echar de menos es *drill-down*. Como ya hemos dicho, se trata de la inversa de *roll-up* y no se puede hacer si no disponéis de los datos detallados.

Ejemplo de secuencia de operaciones

Tenemos el cubo $2 \times 2 \times 2$ siguiente:

Unidades producidas por producto, fábrica y mes	Fábrica del Vallès		Fábrica del Prat	
	Enero 2002	Febrero 2002	Enero 2002	Febrero 2002
Bolígrafos	100.000	110.000	450.000	420.000
Gomas	337.000	473.000	904.000	995.000

Queremos ver, para los mismos meses y productos, los artículos vendidos en Cataluña. Tendríamos que hacer un *drill-across* hacia *Ventas*, pero no es posible hacerlo directamente porque las Dimensiones no coinciden. En primer lugar, nos tenemos que deshacer de la Dimensión Fábricas.

$$A := Roll-up_{Fábrica::All}(\text{"Unidades producidas por Producto, Fábrica y Mes"})$$

A	All	
	Enero 2002	Febrero 2002
Bolígrafos	550.000	530.000
Gomas	1.241.000	1.468.000

Ahora solo tenemos un valor en la Dimensión de fábricas. Por lo tanto, podemos hacer un cambio de base para quedarnos con las mismas cuatro celdas, pero una Dimensión menos.

$$B := CambioBase_{Producto \times Tiempo}(A)$$

B	Enero 2002	Febrero 2002
Bolígrafos	500.000	530.000
Gomas	1.241.000	1.468.000

Puesto que B solo tiene dos Dimensiones y las dos las tenemos en *Ventas*, ya podemos hacer el *drill-across*. Dado que la función que utilizamos es la identidad entre Dimensiones, no hay que explicitarla, sino simplemente decir cuál es el Hecho destino (*Ventas* en este caso).

$$C := Drill-across_{Ventas}(B)$$

C	Enero 2002	Febrero 2002
Bolígrafos	artículos: 465.837	artículos: 513.284
	ingresos: 973.427'30	ingresos: 1.075.143'80

C	Enero 2002	Febrero 2002
Gomas	artículos: 1.348.378	artículos: 1.490.281
	ingresos: 498.462'20	ingresos: 523.093'90

Hemos obtenido ahora celdas de Ventas con todas sus Medidas. Solo estábamos interesados en ver artículos, así que podemos proyectar únicamente esta Medida.

$$D := \text{Proyección}_{\text{artículos}}(C)$$

D	Enero 2002	Febrero 2002
Bolígrafos	465.837	513.284
Gomas	1.348.378	1.490.281

Ahora ya tenemos los datos que deseábamos, pero no queremos que hagan referencia a todos los supermercados, sino solo a los que hay situados en Cataluña. Por lo tanto, hay que introducir la Dimensión geográfica.

$$E := \text{CambioBase}_{\text{Artículos} \times \text{Lugar} \times \text{Tiempo}}(D)$$

E	Estado español	
	Enero 2002	Febrero 2002
Bolígrafos	465.837	513.284
Gomas	1.348.378	1.490.281

Si ahora queremos seleccionar los datos de Cataluña, antes tenemos que conseguir los datos del Estado español detallados por región. En este caso lo podemos hacer, porque tenemos disponibles los datos por población (tal y como indica el esquema de la figura 21). Observad que si los tuviéramos solo almacenados por Estado, no lo podríamos hacer.

$$F := \text{Drill-down}_{\text{Lugar}::\text{Región}}(E)$$

F	Cataluña		Castilla y León		Madrid		Andalucía	
	Enero 2002	Febrero 2002	Enero 2002	Febrero 2002	Enero 2002	Febrero 2002	Enero 2002	Febrero 2002
Bolígrafos	275.827	290.918	85.472	111.291	58.172	59.723	46.366	51.352
Gomas	784.172	918.012	293.829	288.409	141.003	140.298	129.374	143.562

Finalmente, solo hemos de seleccionar los datos de Cataluña para tener lo que nos interesa.

$$R := \text{Selección}_{\text{Región}.nombre = "Cataluña"}(F)$$

R	Cataluña	
	Enero 2002	Febrero 2002
Bolígrafos	275.827	290.918

R	Cataluña	
	Enero 2002	Febrero 2002
Gomas	784.172	918.012

2.3. Restricciones de integridad inherentes al modelo

Ya hemos visto cuáles son los elementos del modelo y las operaciones que se pueden hacer con los datos. Ahora tenemos que ver qué datos no se pueden insertar y qué operaciones no están permitidas.

2.3.1. Unicidad y entidad de la Base

Recordad que un cubo es una función que va de un espacio n -dimensional a una Celda.

A los distintos conjuntos de Niveles que definan espacios en los que podamos colocar las instancias de una Celda los denominaremos Bases.

No es necesario que todas las Dimensiones de una Celda participen en una Base. La Base simplemente indica qué Dimensiones identifican las celdas (es el mismo concepto que la clave candidata del modelo relacional).

Figura 24

VentaAtomica	C
articulos: <i>integer</i> ingresos: <i>real</i>	
<>Base>> [Producto, Dia, Poblacion, Cliente]	

Ejemplo de Base

En nuestro ejemplo de la cadena de supermercados, una cierta venta estaría identificada por un día, un producto, una población y un cliente. Por lo tanto, *Producto*, *Dia*, *Poblacion* y *Cliente* definen un espacio en el que podemos colocar las instancias de VentasAtómicas. En la figura 24, podéis ver cómo representaríamos esto.

Las Bases tienen que cumplir las restricciones siguientes:

- No podemos colocar dos celdas en el mismo punto del espacio. Por lo tanto, los valores que tengan las celdas para las asociaciones con los Niveles de una Base deben ser diferentes para cada celda. Por ejemplo, no puede haber dos instancias de *VentaAtomica* asociadas al mismo *Producto*, *Dia*, *Poblacion* y *Cliente*.

- Tenemos que saber en qué punto del espacio colocamos cada celda. De este modo, las asociaciones con los Niveles de una Base no admitirán el valor nulo. Cada celda tiene que estar relacionada con una instancia de cada Nivel que forma una Base. Las asociaciones con estos Niveles tienen como multiplicidad mínima un 1 del lado del Nivel.
- Los Niveles que forman una Base han de ser funcionalmente independientes. Si hay un Nivel que depende de los otros, lo sacaremos del conjunto que forma la Base. En nuestro ejemplo, no podría ser que un cliente solo pudiera comprar a una cierta población. Si fuera de esta manera, la base estaría formada solo por los Niveles Cliente, Dia y Producto. La población ya quedaría determinada por el cliente.

El conjunto de Niveles que forman una Base tienen que ser funcionalmente independientes. Además, las asociaciones de estos Niveles con la Celda deben tener multiplicidad mínima 1 del lado del Nivel y no es posible que dos celdas estén asociadas con las mismas instancias para todos los Niveles de la Base.

2.3.2. Acumulación o agregación

Sin duda, la operación más característica del análisis multidimensional es el *roll-up*. Por desgracia, también es la más problemática. Veamos cuáles son las tres condiciones para que el resultado de una agregación sea correcto.

Compatibilidad

Habitualmente, cuando se acumulan un conjunto de datos, lo que se hace es sumarlos, pero no siempre es así. Realmente se pueden aplicar otras operaciones, como por ejemplo la media, el mínimo, el máximo o, incluso, el producto. La operación que se aplique depende del tipo de Medida que agreguemos y de la Dimensión a lo largo de la que lo hacemos. Ciertas operaciones son incompatibles con algunos tipos de Medidas y Dimensiones.

Ejemplos de operaciones

La cantidad vendida durante un mes es la suma de las cantidades vendidas cada uno de los días.
O para obtener el interés anual, multiplicamos los mensuales.

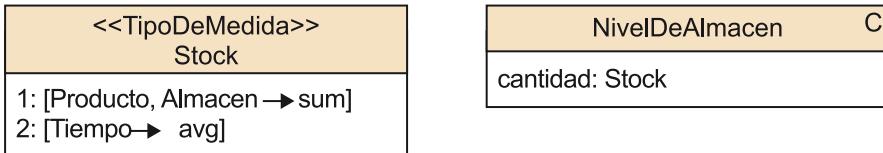
La operación de agregación, el tipo de Medida que agregamos y la Dimensión a lo largo de la que lo hacemos tienen que ser compatibles.

Agregación de stocks

Un ejemplo claro de agregación diferente según la Dimensión son los *stocks* de productos en los almacenes. Si registramos el *stock* a diario en cada almacén que tengamos, el *stock* mensual no será la suma de los *stocks* diarios (la suma es incompatible con los *stocks* y la Dimensión Tiempo). Si hoy tengo un coche en el almacén y mañana también tengo uno, en total no tengo dos, porque realmente son el mismo coche. Lo más habitual en este caso es hacer la media. En cambio, si dispongo de dos almacenes y un coche al mismo tiempo en cada uno de estos, sí tengo dos coches.

La operación de agregación más común es la suma. Por este motivo, en vez de hablar de compatibilidad entre Medida, Dimensión y operación, a veces se habla simplemente de Medidas aditivas (si se pueden sumar en cualquier Dimensión), semiaditivas (si hay Dimensiones en las que no se pueden sumar) y no aditivas (si no se pueden sumar a lo largo de ninguna Dimensión).

Figura 25



Las Medidas

Las Medidas que hacen referencia a ingresos acostumbran a ser perfectamente aditivas a lo largo de cualquier Dimensión. Por el contrario, las que hacen referencia a estados, como por ejemplo los stocks o los saldos, suelen ser semiaditivas (no se pueden sumar a lo largo del tiempo). Finalmente, algunas que hacen referencia a intensidad (como por ejemplo, la temperatura) o se basan en fórmulas matemáticas son no aditivas.

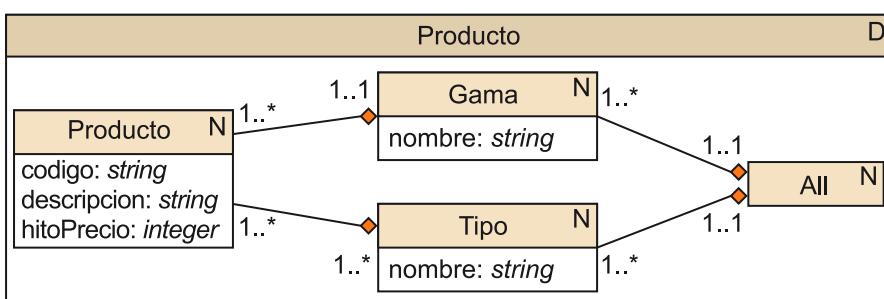
Si no se dice lo contrario, asumiremos la utilización de la suma para hacer *roll-up*. Si quisiéramos utilizar una operación matemática distinta, deberíamos explicitarlo como un parámetro más de la misma operación (*roll-up_{Nivel}(cubo,operación)*) o en el esquema (definiendo un tipo de Medida como muestra la figura 25). En este caso, los stocks se agregarían mediante la suma a lo largo de Producto y Almacen, y mediante la media a lo largo de Tiempo. Dado que ciertas operaciones no son conmutables (no es lo mismo la suma de mínimos que el mínimo de las sumas), también podéis indicar el orden en el que se tienen que hacer las agregaciones (en el ejemplo, primero se harían las sumas y después las medias).

Disyuntividad

Según la operación de agregación que apliquemos, corremos el riesgo de considerar más de una vez el mismo dato sin que nos demos cuenta. Esto sucede cuando hay un Nivel que contiene instancias con conjuntos de partes no disyuntos. En estos casos, es obligatorio indicar las multiplicidades en la jerarquía de agregación para saber qué celdas podemos utilizar en la agregación y cuáles no. En el peor caso, deberemos utilizar las instancias de la Celda atómica, que por definición tienen que ser disyuntas.

Ciertas operaciones de agregación piden que los conjuntos de partes de las instancias que utilizamos como operandos sean disyuntos.

Figura 26



Ejemplo de instancias no disyuntas

Considerad ahora la Dimensión `Producto` como la tenéis dibujada en la figura 26. Se ha añadido un nuevo Nivel: `Tipo`. Ahora resulta esencial explicitar las multiplicidades de las agregaciones. Observad que la agregación entre `Producto` y `Tipo` tiene multiplicidad $1..* - 1..*$ en lugar de $1..* - 1..1$ como el resto de las agregaciones. Esto quiere decir que hay productos que son de más de un tipo y los conjuntos de partes de las instancias de `Tipo` no son disyuntos. El producto "Kinder sorpresa" es un juguete y un chocolate. Por consiguiente, pertenece a los tipos "Juguete" y "Chocolate" al mismo tiempo. Si ahora intentamos calcular los ingresos totales sumando los ingresos de juguetes y chocolates, estaremos contando los ingresos por ventas de "Kinder sorpresa" dos veces. Según el tipo de medida, esto puede ser correcto o no. Podéis indicar los casos en los que no sea válido, como se muestra en la figura 27. De este modo, sabemos que no tenemos que calcular todas las Medidas de tipo `Ingreso` a partir de datos de granularidad `Tipo`, sino que deberemos hacerlo desde una granularidad más pequeña, como por ejemplo `Producto`.

Figura 27

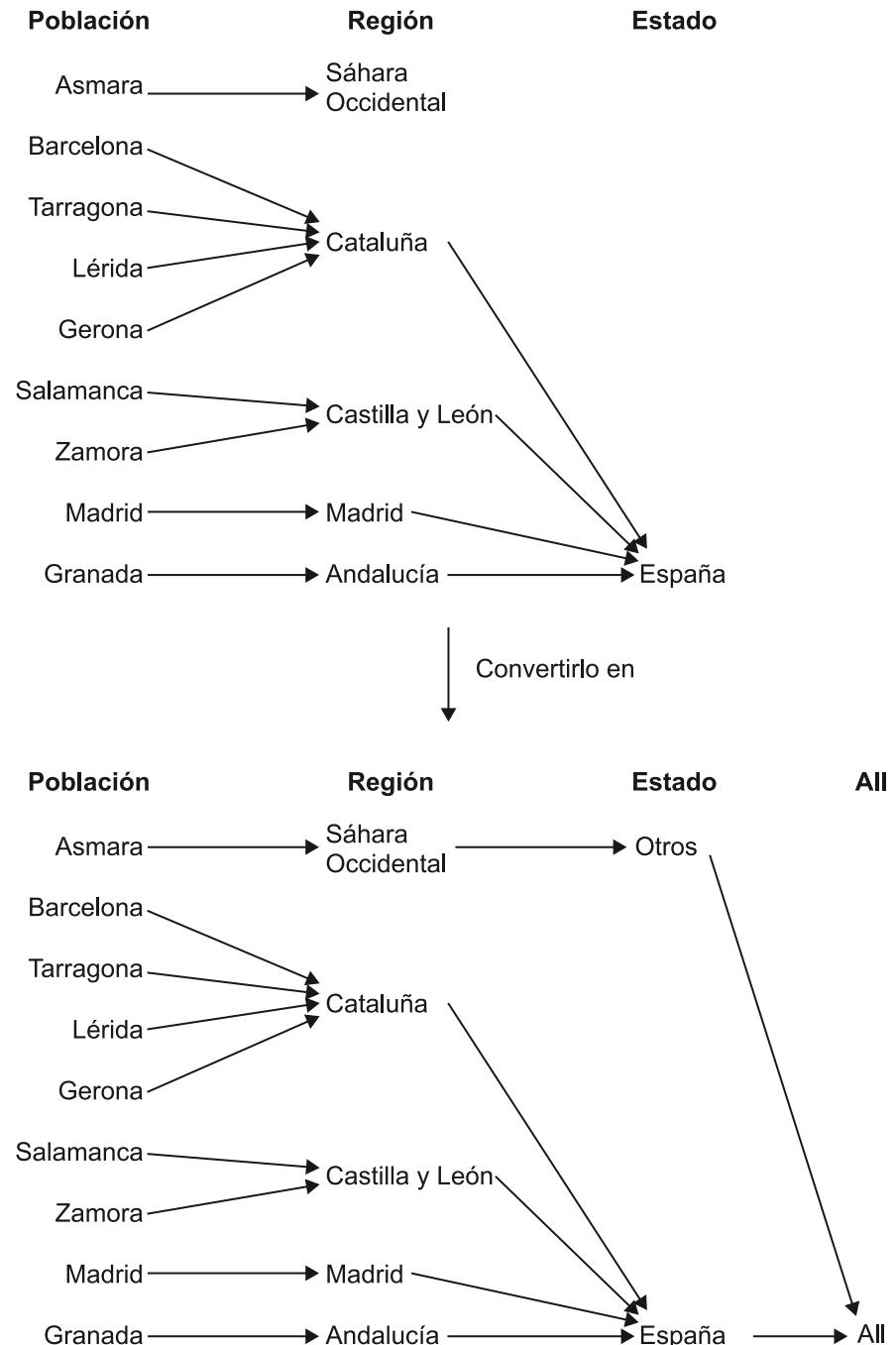
Tipo	N	VentaAtomica	C
nombre: <i>string</i>		articulos: <i>integer</i>	
Fuente inválida por: <code>Ingreso</code>		ingresos: <code>Ingreso</code>	

Compleitud

Podría suceder que algunos Niveles contuvieran instancias que no participaran en la composición de ninguna instancia de los Niveles superiores. Esto lo tenemos que evitar añadiendo instancias ficticias al Nivel superior, para no olvidarnos de operar los datos asociados a estas instancias. La multiplicidad mínima de la agregación junto con el compuesto tiene que ser siempre 1. Un Nivel ha de cubrir completamente los Niveles que tiene por debajo en la jerarquía de agregación.

Tenemos que garantizar que todas las instancias participan como mínimo en una instancia de los Niveles inmediatamente superiores de la jerarquía de agregación.

Figura 28



Ejemplo de no completitud en la jerarquía de agregación

Imaginemos que ampliamos el negocio de supermercados y abrimos uno nuevo en Asmara. Además de añadir esta población a la Dimensión Lugar, también la añadimos a la región "Sáhara Occidental" de la que forma parte. Sin embargo, puesto que el referido de autodeterminación todavía no se ha hecho, decidimos no registrarle a qué Estado pertenece (podéis ver cómo queda la Dimensión Lugar en la parte superior de la figura 28). ¿Cómo podemos calcular ahora el total de ingresos? Ya no sirve consultar los ingresos obtenidos en el Estado español, porque hay una región que no es parte de "Estado español". A pesar de que *Region* cubre completamente *Poblacion*, *Estado* ya no cubre *Region*. La solución es crear una instancia artificial de *Estado*, que podemos denominar "Otras", que tenga como partes "Sáhara Occidental" y todas las otras regiones que no formen parte de ningún Estado. Con esto, *Estado* ya cubre *Region*, pero todavía hay que crear el Nivel *All* con una instancia que agrupe "Estado español" y "Otras" (la Dimensión *Lugar* queda como está dibujada en la parte inferior de la figura 28). Observad

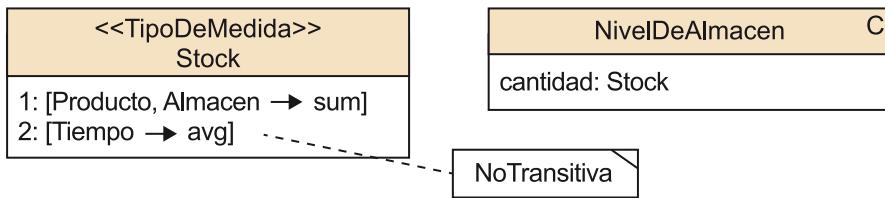
que antes de añadir "Asmara" no hacía falta el Nivel All, porque el Nivel más alto de la jerarquía ya solo tenía una instancia.

2.3.3. Transitividad

La última restricción que debemos tener en cuenta, también relacionada con la agregación, es la transitividad de las operaciones de agregación. A veces, no obtenemos el mismo resultado si operamos con resultados parciales que si operamos directamente con los datos básicos. Hay operaciones que no son transitivas (por ejemplo, la media).

Tenemos que asegurarnos de que las operaciones de agregación son transitivas. En caso de que no lo sean, el resultado correcto siempre es el que se obtiene de operar con los datos atómicos.

Figura 29



Ejemplo de no transitividad

Por simplicidad, imaginemos que abrimos el supermercado cada día del año. Cada día del mes de enero tenemos en el almacén 100 unidades, cada día del mes de febrero 300 y cada día del mes de marzo, 200. Parece claro que la media de unidades que tenemos en enero es 100, en febrero, 300 y en marzo, 200. ¿Cuál es el stock durante el primer trimestre? ¿200 unidades? No podemos hacer la media de las medias mensuales. Tenemos que hacer la media de los stocks diarios. Realmente, la media es 196,6 ($100 \times 31 + 300 \times 28 + 200 \times 31 = 196,6$). Si interesa reflejar esto, lo podemos hacer como podéis ver en la figura 29.

3. Diseño conceptual

A pesar de que las herramientas OLAP se consideran relativamente fáciles de utilizar, construirlas y mantenerlas requiere conocimientos especializados. Por este motivo, dedicaremos buena parte de este módulo a estudiar su diseño. Del mismo modo que para las bases de datos operacionales, lo haremos en tres pasos: diseño conceptual, diseño lógico y diseño físico. En este apartado veremos cómo se puede hacer un diseño conceptual multidimensional.

Figura 30



Lo primero que haremos es conseguir un esquema expresado con UML de los cubos de datos que nos interesan.

Un Hecho y su correspondiente conjunto de Dimensiones forman una Estrella.

Un conjunto de Estrellas forman una Constelación.

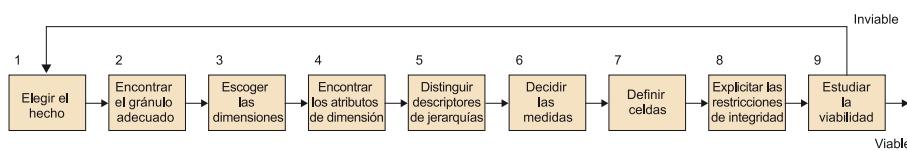
Podemos representar una Estrella como un paquete (como podéis ver en la figura 30). Veamos ahora cómo podemos diseñar lo que hay dentro de cada uno de estos paquetes.

3.1. Una metodología para diseñar una Estrella

Una de las ventajas del diseño multidimensional es que utiliza la técnica de “divide y vencerás”. No intentamos diseñar de golpe todos los almacenes departamentales de nuestra empresa, ni satisfacer al mismo tiempo las necesidades de los analistas, sino que hacemos pequeños diseños bastante independientes unos de otros.

Diseñamos por separado cada Estrella y lo hacemos en un proceso iterativo que consta de nueve pasos.

Figura 31



3.1.1. Elegir el Hecho

Antes que nada, tenemos que saber qué tema estudiaremos con nuestra Estrella: cuál será el Hecho. Un Hecho no es más que un conjunto de acontecimientos con datos numéricos asociados. Los candidatos habituales son los procesos de negocio. Por **procesos de negocio** entendemos todos los procesos operacionales de la organización que están soportados por algún sistema informático del que se pueden extraer datos. Probablemente, los mismos analistas os sugerirán el proceso de negocio que tenéis que modelar. Sin embargo, os podéis dar cuenta del condicionamiento tan fuerte que tenemos para la elección de un Hecho: debemos disponer de los datos, preferiblemente en soporte informático.

Lo primero que debemos hacer para diseñar una Estrella es elegir el Hecho objeto de análisis.

En primer lugar, intentad definir Hechos "pequeños". Definir de golpe una única Estrella que incluya todos los ámbitos de la empresa es prácticamente imposible. Lo más fácil es empezar desarrollando Estrellas para analizar procesos de negocio que estén soportados por un único sistema operacional. Por ejemplo, si tenemos un sistema de gestión de ventas, nos resultará relativamente fácil diseñar y poner en funcionamiento una Estrella para el análisis de ventas.

3.1.2. Encontrar el gránulo oportuno

El gránulo es el individuo último que queremos analizar, la Celda más pequeña que queremos tener disponible.

Ya no hablamos de un concepto grande y abstracto como un proceso de negocio, sino del tipo de objeto concreto que queremos analizar. Los gránulos más típicos son las transacciones individuales (por ejemplo, las ventas) o las instantáneas diarias de un estado (por ejemplo, los stocks). Generalmente, suele ser cualquier tipo de acontecimiento que se dé con una frecuencia relativamente alta.

Ejemplo de gránulos posibles

Los gránulos candidatos a nuestra Estrella de ventas podrían ser "ventas mensuales de producto por tienda", "ventas diarias de un cierto producto a un cliente en una población", "compras que hace un cliente en un momento dado en una cierta tienda" o, incluso, "líneas de los tiqués de compra", que representan la venta de un producto en un momento dado en una cierta tienda.

Elegir el gránulo de nuestro Hecho es muy importante, porque determina la dimensionalidad de la base de datos y, como veremos cuando estudiemos la viabilidad de la implementación, tiene un impacto directo en el tamaño del conjunto de datos. Un gránulo muy pequeño resultará en una base de datos muy grande. Por el contrario, un gránulo más grande implicará renunciar a alguna Dimensión o, más posiblemente, a calcular ciertas Medidas, a causa, por ejemplo, de problemas con la transitividad de las operaciones o la no disyuntividad de los compuestos.

Ejemplo de elección del gránulo

¿Qué ocurre si elegimos como gránulo "ventas mensuales de producto por tienda"? ¿Cómo sabremos qué día del mes hemos vendido más? Este sería un gránulo demasiado grande. Ahora bien, pensad los problemas que puede generar elegir como gránulo "líneas de boletos de compra". Cada vez que una cajera pasa un artículo por el lector de códigos de barras, generamos una celda en nuestro sistema. ¿Cuánto tarda una cajera en hacer esto? ¿Cuántas cajeras tenemos? ¿Cuánto tiempo trabajan a diario? Probablemente, el sistema acabará contenido más datos de los que puede manejar. Si realmente no nos hace falta tanto detalle, deberíamos intentar reducir el volumen de datos de alguna manera. En nuestro ejemplo, pensamos que los sistemas operacionales identifican a cada cliente (por ejemplo, con el número de tarjeta de crédito) y que un mismo cliente compra el mismo artículo muchas veces, de modo que podemos generar una única celda para todas sus compras de un mismo producto, en todo un día, dentro de la misma población.

Un buen diseño siempre pide elegir el gránulo más pequeño posible, no porque los usuarios quieran ver siempre los datos más detallados, sino para no perder la posibilidad de calcular ningún dato derivado con el mínimo error. El límite siempre es la disponibilidad de datos en los sistemas operacionales. No podemos elegir "líneas de los tiqués de compra" si el sistema operacional correspondiente solo registra la cantidad total de la compra.

Elegir un gránulo demasiado grande representa perder información. Sin embargo, elegirlo **demasiado** pequeño puede representar derrochar espacio o llegar a hacer inviable el proyecto por exceso de datos.

Tamaño de un gránulo

Entendemos que un gránulo es más grande cuanto más acontecimientos representa. De este modo, es más grande la granularidad Mes que Día y Día que Hora.

Ved también

Ya hemos visto estos conceptos en el apartado "Restricciones de integridad inherentes al modelo".

3.1.3. Elegir las dimensiones que se utilizarán en el análisis

Unas dimensiones típicas son Tiempo, Producto, Cliente, Promoción, Almacén, Tipo o Estado. Si el gránulo está claramente definido, encontrar un primer conjunto de Dimensiones de análisis es inmediato a partir de la misma definición. A este primer conjunto inicial le podemos añadir otras Dimensiones, pero solo si cada combinación de las instancias de las Dimensiones inicia-

les determina una instancia de la Dimensión que queremos añadir. Si determinara más de una, significaría que tenemos que reconsiderar la Dimensión o el gránulo mismo para poder añadirla.

La multiplicidad de las asociaciones entre Hechos y Dimensiones siempre es muchos a uno. Muchas instancias del Hecho se asocian con la misma instancia de Dimensión, pero solo una instancia de Dimensión se asocia con cada instancia del Hecho.

Ejemplo de elección de dimensiones

Si nuestro gránulo es "ventas diarias de un cierto producto a un cliente en una población", entonces tendremos al menos las Dimensiones Tiempo, Producto, Cliente y Lugar. Además de estas cuatro Dimensiones, hay otras, como por ejemplo Promoción, que quedan determinadas por el resto. Un cierto producto en un momento dado y en una población está siendo promocionado de alguna manera (por ejemplo, con un 10% de descuento) y queremos estudiar cómo afecta esto a las ventas. Por el contrario, no podríamos añadir la Dimensión Publicidad porque no podemos asociar ningún tipo de publicidad en concreto a una celda. Imaginemos que hay carteles publicitarios en las carreteras, que coinciden con diferentes campañas radiofónicas locales y otras campañas televisivas estatales. ¿Cómo podemos asociar esto a una venta en una población? Si lo que se anuncia es un establecimiento, ¿cómo lo asociaremos a la venta de un producto? Las compras pueden no coincidir exactamente en el tiempo con las campañas publicitarias que las provocan. ¿Cómo asociaremos una campaña a un día en concreto? Publicidad no queda determinada por Tiempo, Producto, Cliente y Lugar.

El gránulo mismo ya determina un primer conjunto de Dimensiones. Tenemos que añadir a este los otros puntos de vista que queramos utilizar en el análisis y que queden determinados por el conjunto inicial de Dimensiones.

Lo más habitual es encontrar entre cuatro y quince Dimensiones. Encontrar dos o tres es muy extraño y os tendría que hacer sospechar que hay alguna Dimensión más que se podría añadir al diseño (por ejemplo, la Dimensión temporal es casi omnipresente). En el extremo opuesto, que haya veinte tampoco parece fácilmente justificable. Muchas de estas Dimensiones resultarán prescindibles o encontraremos algunas parejas o grupos de las mismas que estarán fuertemente correlacionados, de modo que se podrían representar como una única Dimensión, como veremos en el apartado "Dependencias entre Dimensiones".

Las dimensiones

Tenéis que entender las dimensiones como las variables independientes que afectan a cada observación. Pensad que resulta difícil encontrar un fenómeno con muchas variables y uno con pocas no es demasiado interesante de analizar.

3.1.4. Encontrar los atributos de cada dimensión

De entre los atributos que podemos encontrar en los sistemas operacionales, debemos elegir los que pertenecen a las Dimensiones y nos serán útiles para elegir y describir el espacio de análisis. Tenemos que seleccionar cualquier atributo que creamos que pueda ser útil para seleccionar, agrupar o simplemente

poner como cabecera de un informe. Una Dimensión puede tener con relativa facilidad más de cincuenta atributos. Hay que documentar tanto el origen como la interpretación de cada uno de los atributos.

Dado que los atributos de Dimensión tienen que servir para hacer selecciones y agrupaciones, han de tener siempre un dominio discreto, nunca continuo. ¿Cómo seleccionaréis un valor si tiene infinitos decimales? Además, un atributo nunca tiene que estar codificado ni abreviado y debe ser fácilmente inteligible para el usuario (en principio, un código de barras no es un buen atributo para la Dimensión Producto). Puede interesar tener algunos códigos no directamente interpretables (como por ejemplo el código de barras) para identificar las instancias dentro del mismo sistema. No obstante, dado que nunca serán utilizados por el usuario, tenemos que calcular muy bien el espacio que podemos ahorrarnos. Finalmente, también hay que decir que la calidad de un atributo debe estar garantizada: no puede haber errores en su escritura.

Cuando se definieron las Dimensiones dentro del apartado "Estructuras de datos", en "Componentes del modelo multidimensional", ya se habló de atributos de Dimensión y de las características más importantes. Sin embargo, a modo de recordatorio, tenemos la definición siguiente:

Un atributo de Dimensión debe estar definido sobre un dominio discreto, y tiene que ser descriptivo, fácil de recordar y comprensible a primera vista. Los atributos textuales suelen pertenecer a las Dimensiones, mientras que los atributos numéricos son habitualmente Medidas.

Ejemplo de Dimensión temporal

Por ejemplo, en una Dimensión temporal, además de saber el día de la semana, el mes o el año, tendríamos que poder seleccionar días festivos o laborables, semanas de vacaciones o de clase, de matrícula o de exámenes, años normales o bisiestos, etc.

Ejemplo de atributo de Dimensión

Podremos considerar que el precio de un producto (un atributo numérico) nos viene dado y que nunca cambiará. Si creemos que nos puede ser útil para seleccionar ventas, estaríamos hablando de un atributo de la Dimensión Producto en la Estrella Ventas.

Si un atributo nos parece interesante y no cumple las características deseables, es preciso que lo modifiquemos hasta que tome la forma adecuada: haciéndolo más explicativo (por ejemplo, cambiando códigos por frases), definiendo rangos de valores (por ejemplo, precios de menos de un euro, de entre uno y cinco euros, de entre cinco y cincuenta o de más de cincuenta), limpiándolo (por ejemplo, eliminando valores nulos o arreglando errores de ortografía), etc.

Se tiene que convertir cualquier tipo de código que haya en los sistemas operacionales en un texto explicativo y discretizar los dominios continuos para facilitar la selección y comprensión de los atributos de Dimensión.

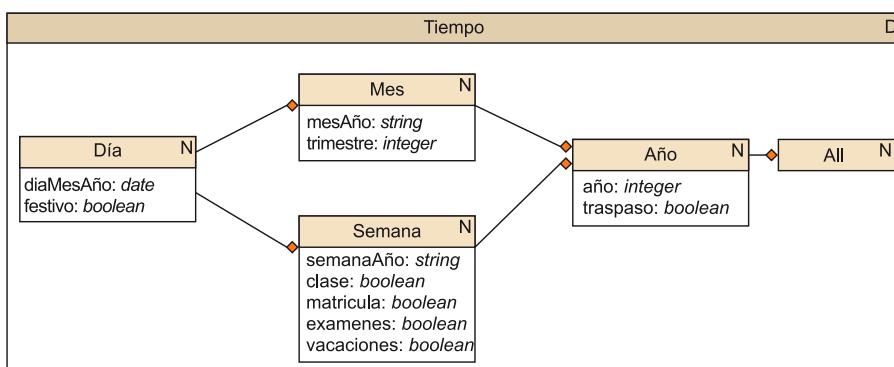
3.1.5. Distinguir entre descriptores y jerarquías de agregación

De entre los atributos que hay en una Dimensión, tenemos que distinguir dos tipos: los que utilizaremos para agrupar y los que servirán simplemente para seleccionar. Por ejemplo, utilizaremos los meses y los años para agrupar días, mientras que usaremos la festividad o no festividad de un día simplemente para seleccionar. Los primeros son los que definen las jerarquías de agregación, y los segundos son los descriptores.

Un atributo por sí mismo no es de un tipo o de otro. Que un atributo defina un Nivel dentro de una jerarquía de agregación o no depende mucho más de lo que los usuarios desean, que de los datos en sí mismos. Las jerarquías se tendrían que consensuar con los usuarios. Con jerarquías demasiado grandes aumentamos la complejidad del sistema, pero con jerarquías demasiado simples podemos perder posibilidades de análisis o retardar el sistema.

El Nivel atómico de las jerarquías queda definido por el gránulo que hayamos elegido. Si tenemos ventas diarias, el Nivel atómico de la Dimensión Tiempo será Día y no Hora ni Mes. A partir de este Nivel, el resto del grafo está definido por las dependencias funcionales que haya entre los atributos de agrupación que elegimos. Colocaremos el resto de los atributos como Descriptores en el Nivel que les corresponda.

Figura 32



Otra interpretación de la jerarquía de agregación

Si volvemos a observar ahora la Dimensión de la figura 14, podemos ver que Día determina Semana y Mes, pero ni Semana determina Mes ni Mes determina Semana, a pesar de que los dos determinan Año. Recordad que las dependencias funcionales cumplen la propiedad transitiva. Por consiguiente, dado que Mes determina funcionalmente Trimestre y este determina Año, podemos decir que Mes determina Año. Sin embargo, recordad también que las agregaciones cumplen igualmente la propiedad transitiva y que no hay que explicitar la que hay entre Mes y Año. Si los usuarios no utilizaran los trimestres para agrupar meses, este Nivel desaparecería de la jerarquía (sería simplemente un atributo asociado a Mes, como podéis ver en la figura 32), y entonces sí que dibujaríamos la agregación entre Mes y Año. En esta figura, hemos dibujado en cada Nivel solo los Descriptores que dependen directamente del mismo.

Podéis entender las agregaciones que forman una jerarquía de agregación como dependencias funcionales entre atributos de Dimensión que utilizamos para agrupar.

3.1.6. Decidir cuáles son las medidas que interesan

De manera típica, las Medidas son atributos numéricos que se pueden sumar, como por ejemplo las cantidades vendidas o los ingresos que genera una venta. Podéis elegir cualquier atributo que tengáis directamente disponible en las bases de datos operacionales o que podáis derivar a partir de los que tengáis. Pensad solo que el espacio que ocupen las Medidas (es decir, el tamaño de las celdas) afectará sensiblemente al volumen total de la base de datos (si ocupan mucho, la base de datos será demasiado grande para ser funcional). Lo mejor es encontrar todas las Medidas interesantes para los analistas y después elegir solo las que pensemos que serán más útiles.

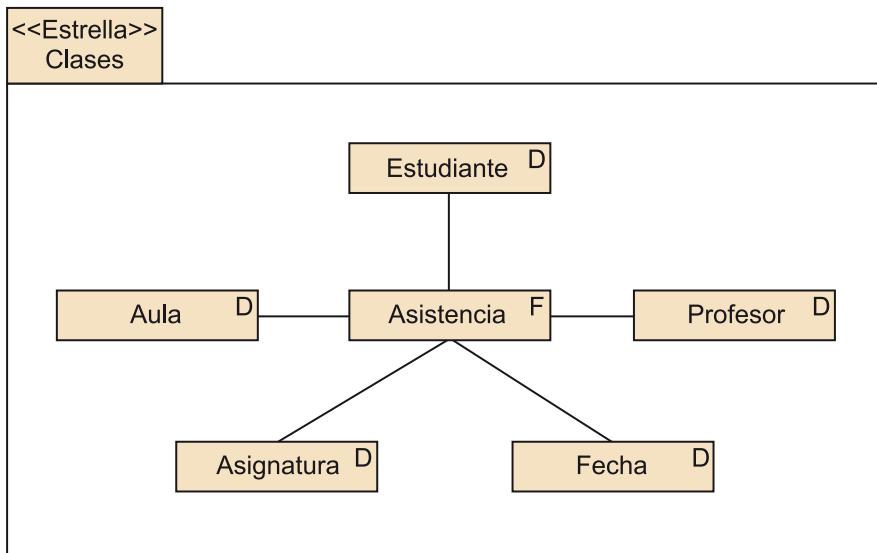
Nota

Si una Medida es derivada, es preciso que lo indiquéis como haríais con cualquier atributo con UML (poniendo una barra delante del nombre y adjuntando un comentario con la fórmula utilizada en la derivación).

Las Medidas son atributos numéricos normalmente aditivos.

Generalmente, cuantas más Medidas contenga un Hecho, más útil será la Estrella. Sin embargo, a veces encontraréis algunos Hechos muy útiles que no tienen ninguna Medida. Como ya hemos dicho antes, las instancias de los Hechos pueden representar acontecimientos. Normalmente, al darse los acontecimientos, medimos algo. Sin embargo, hay casos en los que únicamente queremos tener constancia de que el acontecimiento se ha dado. En este caso, el Hecho no tiene ninguna Medida.

Figura 33



Ejemplo de Hecho sin Medidas

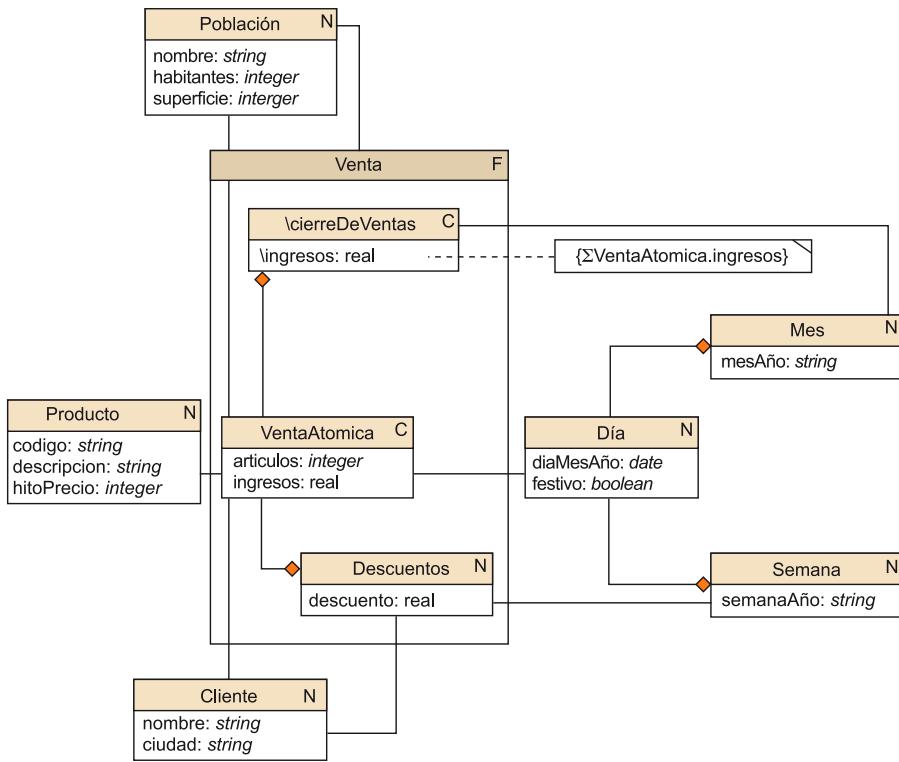
Pensad que estamos en una universidad presencial y queremos analizar la asistencia a clase de cada alumno, según la asignatura, el profesor que imparte la clase y el aula y la fecha en los que se hace esta. Tendríamos la Estrella de la figura 33. Asistencia no tendría ninguna Medida, porque lo único que nos interesa es tener constancia de si un alumno ha asistido a clase o no. Con esta Estrella sin Medidas, ya podemos contestar consultas, como por ejemplo qué profesor tiene más alumnos en clase o cuántos profesores han hecho alguna clase en una determinada aula.

Hay algunos Hechos que no tienen ninguna Medida.

3.1.7. Definir Celdas

Veréis que en el conjunto de Medidas que habéis elegido hay diferentes granularidades. Cada una pertenece a una de las Celdas del Hecho y tenéis que ir con mucho cuidado para ponerlas allí donde les corresponde. Tenéis que dibujar Celdas de distinta granularidad dentro del mismo Hecho, para contener las Medidas de la granularidad correspondiente. De todo el retículo de Celdas que tenéis ejemplificado en la figura 20, solo es preciso que dibujéis las Celdas que contienen Medidas o que consideréis especialmente relevantes.

Figura 34



Ejemplo de Medidas de diferente granularidad

Pensemos que nos interesan cuatro Medidas: el número de artículos vendidos, los ingresos que genera una venta, los ingresos que tenemos mensualmente en cada población (cuando cerramos las cuentas) y el descuento que hacemos a cada cliente. Consideraremos que el descuento se aplica de manera semanal y depende del gasto que ha hecho este cliente durante el periodo de tiempo correspondiente y del trato que tengamos con el mismo. En la figura 34, podéis ver cómo quedaría el esquema. Aparecen tres Celdas distintas que representan las tres granularidades que nos interesan. Cada Celda contiene las Medidas que tenemos en esta granularidad. Observad que los ingresos mensuales por

población son una Medida derivada que se obtiene sumando los ingresos que tenemos en Celdas de granularidad más pequeña. Puesto que se trata de la única Medida que contiene esta Celda, también podemos marcar la Celda como derivada (con una barra delante del nombre). Por el contrario, no podemos calcular el descuento directamente de las otras Medidas, porque depende de la oferta que hacemos a cada cliente en cada momento. Para simplificar la representación, podemos considerar que, si una Celda no está asociada a ningún Nivel de una Dimensión, lo está al Nivel All. De este modo, a pesar de que no quede reflejado en el dibujo, Descuentos está asociado al Nivel Poblacion::All y al Nivel Producto::All.

De todas las Celdas que explicitéis, será necesario almacenar algunas de manera física y otras solo contendrán Medidas que se podrán obtener simplemente haciendo *roll-up* y aplicando la operación que corresponda (y que marcaremos como derivadas). Sin embargo, en el diseño conceptual no tenemos que considerar si unos datos derivados se almacenan físicamente o simplemente se calculan cuando es necesario. Esto ya será una decisión de diseño físico. En este momento, solo hay que dejar constancia de todo lo que se considere especialmente relevante.

Hay que explicitar las Celdas que contienen Medidas interesantes para los analistas, aunque sean derivadas. Podríamos no poner todos los elementos que sean derivados en el esquema. Los explicitamos solo para remarcar lo importantes que son.

3.1.8. Explicitar las restricciones de integridad

Una vez ya tenemos todas las Medidas, Celdas y Niveles, tan solo queda expresar las restricciones de integridad correspondientes (Bases, restricciones de agregación y restricciones de transitividad), como ya hemos visto en el apartado "Restricciones de integridad inherentes al modelo".

Como último paso del diseño conceptual, hay que expresar las restricciones de integridad.

Como especialmente importantes, en este momento hay que destacar las Bases. El conjunto inicial de Dimensiones que encontramos después de definir el gránulo tiene que dar lugar a una Base. Si sustituimos Dimensiones en esta según las dependencias funcionales que hay con el resto de las Dimensiones, podemos obtener las otras Bases del espacio.

Ejemplo de Bases de un espacio

Ya hemos visto antes que una base de VentaAtomica es {Producto, Dia, Cliente, Poblacion}. En el caso de añadir la Dimensión Promocion, esta no formaría parte de la Base, porque queda completamente determinada por las Dimensiones Tiempo, Lugar y Producto (en un momento dado, en una cierta población, un producto solo se promociona de una manera). Si pensamos que una promoción determina un producto y que en todo momento todos los productos tienen una promoción u otra (también podemos considerar la no promoción como un tipo de promoción), entonces podemos sustituir Producto por Promocion y obtener una segunda Base del espacio {Promocion, Dia,

Cliente, Población}. La Base de Descuentos sería {Cliente, Semana} y la de CierreDeVentas sería {Poblacion, Mes}.

3.1.9. Estudiar la viabilidad

Una vez ya hemos acabado el diseño conceptual, hay que ver si la Estrella es realmente implementable o no. Se tiene que estimar el espacio que ocuparán todos los datos. La manera de saber cuánto espacio ocuparán consiste en mirar el contenido de los sistemas operacionales. Si esto es demasiado complicado, podemos hacer una estimación. Para hacerlo, podemos considerar solo lo que ocupará almacenar las instancias del Hecho. El volumen de datos que puedan ocupar las Dimensiones resulta en general insignificante respecto a lo que ocupará el Hecho.

Para hacer la estimación de lo que será necesario para almacenar todas las instancias de una Celda, calculamos el tamaño del espacio que define cada una de sus Bases, y multiplicamos el número de instancias de cada Nivel que las forman. El espacio más pequeño nos dará el número máximo de instancias que puede tener la Celda.

Ejemplo de estimación de instancias

Pensemos que tenemos 1.000 clientes y 1.000 productos distintos, vendemos a 10 poblaciones y queremos almacenar datos durante 1.000 días. Además, tenemos una media de tres promociones diferentes para cada producto (31.000 promociones en total). Si tomamos la Base {Producto, Dia, Cliente, Poblacion}, obtenemos un espacio de $1.000 \times 1.000 \times 1.000 \times 10$ (10^{10} posibles celdas). Por el contrario, si consideramos la Base {Promocion, Dia, Cliente, Poblacion}, obtenemos un espacio de $3.000 \times 1.000 \times 1.000 \times 10$ ($3 \cdot 10^{10}$ posibles celdas). De este modo, el número máximo de celdas que podemos tener en nuestro espacio es 10^{10} .

Este hito puede resultar un poco excesivo. Se tiene que intentar afinar más, conociendo lo dispersas que quedan en este espacio las celdas que queremos analizar en el mismo.

Ejemplo de mejora en el hito

¿Un cliente comprará en todas nuestras tiendas? La respuesta, probablemente, es que no. Comprará solo a las que tenga más cerca de casa. No se darán todas las combinaciones cliente-población y, si se dieran, no sucedería todos los días. Podemos considerar que un cliente solo comprará en una población. De esta manera, a pesar de que el espacio sea de $1.000 \times 1.000 \times 1.000 \times 10$, sabemos que tendremos $1.000 \times 1.000 \times 1.000$ celdas como mucho. Si además estimamos que la media de artículos distintos que compra un cliente al día es 10, nos quedan aproximadamente $10 \times 1000 \times 1000$ celdas.

Una vez sabemos cuántas celdas tendremos, multiplicamos esta cifra por el número de bytes que ocupará cada celda (6 bytes de las Medidas artículos e ingresos más 20 bytes de los identificadores de las 5 Dimensiones, en nuestro caso) y obtendremos una estimación de lo que ocupará nuestra Celda (26×10^7 bytes = 26×10^4 kbytes = 260×10^1 Mbytes = 2,6 Gbytes). Generalmente, lo que ocupa la Celda atómica son órdenes de mayor magnitud que lo que ocupan las otras Celdas (Descuentos ocuparía aproximadamente 1,7 Mbytes

y CierreDeVentas, solo 4,3 kbytes). Solo si tenemos Dimensiones muy pequeñas o muchas Celdas en nuestro esquema, realmente hay que sumar estas cantidades.

Para saber cuánto espacio ocupará nuestra Estrella, lo más realista es observar los datos que contienen los sistemas operacionales (nuestras fuentes de datos) para saber cuántos contendrá en nuestro sistema de análisis. Si no es posible, también lo podemos estimar a partir del número de instancias del Nivel atómico de cada Dimensión y el tamaño de la celda.

¿Será capaz de manejar este volumen de datos nuestro sistema? ¿Será lo bastante buena la velocidad de respuesta? ¿Podemos cargar dentro de la ventana de actualización todos los datos necesarios para mantenerlo actualizado? Si la respuesta a cualquiera de estas preguntas es que no, solo tenemos dos opciones: 1) mejorar el software y el hardware que tenemos o 2) replantear el diseño eligiendo un gránulo no tan fino o descartando algunas Medidas que no sean esenciales.

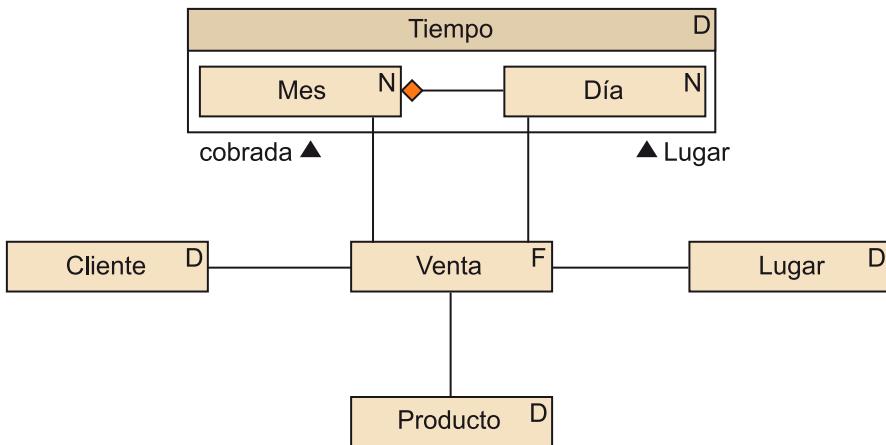
3.2. Reconsideraciones en el diseño conceptual

En este apartado veremos algunas mejoras que, en algunos casos, se pueden hacer en el diseño de una Estrella.

3.2.1. Dimensiones con múltiples roles

Hasta ahora, todas las Dimensiones que hemos utilizado para analizar un Hecho eran muy distintas. Sin embargo, es bastante habitual que haya Dimensiones tan parecidas que, de hecho, deberíamos hablar de la misma Dimensión repetida, que tiene roles diferentes. A veces, de una Dimensión a otra, solo cambia la jerarquía de agregación que tiene definida o los Descriptores que nos resultan interesantes. Normalmente, nos quedaremos con una sola Dimensión y la asociaremos con el Hecho tantas veces como sea necesario. Sin embargo, en algunos casos -por ejemplo, por motivos de confidencialidad o limitaciones en la herramienta OLAP-, podría interesar mantener las Dimensiones diferenciadas.

Figura 35



Ejemplo de Dimensión con roles múltiples

Si en nuestro sistema de ventas admitimos el pago atrasado (a treinta, sesenta y noventa días, por ejemplo), tendremos dos fechas asociadas a cada venta, primero aquella en la que se ha producido la venta realmente y después aquella en la que hemos cobrado. Esto podemos representarlo como veis en la figura 35. Podría ocurrir que las dos asociaciones fueran tanto al mismo Nivel como a Niveles diferentes. Si lo único que nos interesa fuese el mes en que hemos cobrado, tendremos que el final de la asociación `cobrada` se relaciona con el Nivel Más en lugar de Día, pero la Dimensión sigue siendo la misma.

Una Dimensión puede estar asociada más de una vez con un mismo Hecho y tener roles distintos.

3.2.2. Dependencias entre dimensiones

El cociente entre el número máximo teórico de celdas que puede llegar a haber y las celdas que hay realmente es la dispersión del cubo.

Una dispersión muy grande puede generar problemas en la gestión de los cubos. La causa de la dispersión es que ciertas combinaciones de instancias de las Dimensiones no son válidas. El posible valor de una de las Dimensiones depende de los valores de las otras. Cuanto más dependan unas Dimensiones de las otras, más alta será la dispersión.

Si el valor de una Dimensión depende del valor de otra, decimos que están correlacionadas.

Cuando vemos que dos Dimensiones están correlacionadas, las podemos juntar para obtener una sola Dimensión. Esta nueva Dimensión tendrá una instancia para cada elemento del producto cartesiano de las Dimensiones origi-

nales que realmente tenga sentido. A pesar de que somos capaces de representar la misma información, al hacer esto reducimos el número de Dimensiones, pero como contrapartida las que quedan son más grandes.

La correlación entre dos Dimensiones no viene de estas, sino del conjunto de datos que analizamos. En una Estrella podemos ver que dos Dimensiones están muy correlacionadas, mientras que en otra no lo están en absoluto.

Figura 36

	Azul	Rojo	Amarillo
Coche	X	X	
Camión	X		
Tractor			X

Dos dimensiones
Dispersión 9/4

Coche azul	X
Coche rojo	X
Camión azul	X
Tractor amarillo	X

Una dimensión
Dispersión 4/4

Ejemplo de Dimensiones correlacionadas

Imaginemos que hablamos de un concesionario de vehículos, que está interesado en analizar las ventas según el tipo de vehículo y los colores. Este concesionario es muy particular, puesto que, como tenéis esquematizado en la figura 36, solo vende coches rojos o azules, camiones azules y tractores amarillos. El color depende mucho del tipo de vehículo. Por lo tanto, si sus Dimensiones de análisis son Vehículo y Color, tiene un espacio de tamaño nuevo que solo contendrá cuatro celdas. Por el contrario, si tuviéramos una única Dimensión VehículoColoreado, que solo tiene instancias para las parejas vehículo-color que realmente hay, la dispersión sería mínima (habría una celda en cada punto del espacio). Antes teníamos dos Dimensiones de tamaño tres y ahora solo tenemos una, pero de tamaño cuatro.

Tenemos que fusionar Dimensiones solo cuando estén realmente correlacionadas y nos represente una ganancia en la dispersión del cubo sin que el número de instancias de las Dimensiones crezca de manera excesiva. Pensad que los usuarios utilizarán las Dimensiones para seleccionar los datos que quieren ver. Cuantas más instancias tengan las Dimensiones y más complejos sean los conceptos que representan, más difícil les resultará hacer selecciones.

Ejemplo de Dimensiones que no interesa fusionar

En nuestra Estrella de ventas, hemos visto que muy probablemente un cliente siempre compraría en la población donde reside y que esto generaba cierta dispersión en el cubo. Sin embargo, si juntamos las Dimensiones Cliente y Poblacion, no obtenemos ningún concepto concreto. Ningún analista querrá consultar parejas cliente-población. Querrá consultar las ventas a un cliente y las ventas en una población, mayoritariamente por separado. Además, ningún cliente tiene prohibido comprar en una población. Por consiguiente, si las poblaciones son lo bastante cercanas, es posible que el espacio Cliente × Poblacion no sea muy disperso y que todo cliente haya comprado alguna vez a cada una de las poblaciones. La Dimensión ClientePoblacion tendrá una instancia por casi cada elemento del producto cartesiano entre Cliente y Poblacion.

Si queremos reducir la dispersión, lo que tenemos que hacer es fusionar las Dimensiones que estén muy correlacionadas. Esto aumentará el tamaño de las Dimensiones, pero reducirá la dispersión del cubo.

3.2.3. Minidimensiones

En algunos casos, podemos observar que una Dimensión es demasiado grande. Por un lado, esto puede provocar dispersión en el cubo, a menos que el Hecho también tenga muchas instancias; y por el otro, dificulta las consultas. Para solucionarlo, podemos crear una Dimensión más pequeña solo con los atributos más utilizados. A menudo, con esto no basta y lo que tenemos que hacer es, además, modificar el dominio de alguno de estos atributos para que el atributo tenga menos valores posibles.

Tener esta minidimensión no implica eliminar la Dimensión original. Pueden coexistir las dos dentro de la misma Estrella, pero deben aparecer vinculadas por una asociación.

Ejemplo de minidimensión demográfica

Algunas empresas tienen millones de clientes (pensad, por ejemplo, en telefonía). En estos casos, la Dimensión Cliente es excesivamente grande y dificulta de manera clara las consultas. Para solucionar este problema, se definen perfiles de clientes. Se toman solo los atributos más utilizados en las consultas (por ejemplo, edad, sexo, estado civil, nivel adquisitivo, etc.) y se crea una Dimensión demográfica con estos. Si se mantiene la Dimensión Cliente junto con la demográfica, se tiene que definir una asociación entre estas que vincule a cada cliente con su perfil. Si esta Dimensión demográfica todavía tiene demasiadas instancias, podemos crear rangos de valores más pequeños en alguno de los atributos. Por ejemplo, podemos hablar solo de tres valores del atributo edad: edades entre 0 y 30, 31 y 60 y más de 60.

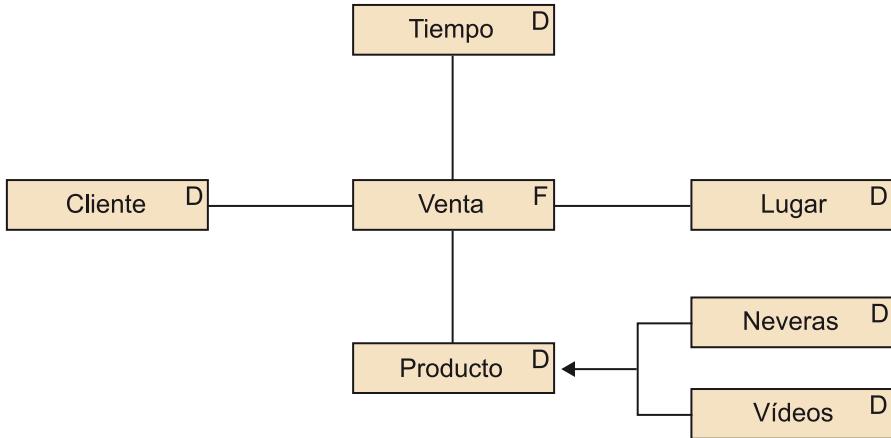
Cuando tenemos Dimensiones demasiado grandes podemos crear minidimensiones con los atributos más utilizados para facilitar las consultas de los usuarios.

Observad que el número de instancias de una minidimensión (como por ejemplo la demográfica) no depende del número de instancias de la Dimensión (por ejemplo, Clientes), sino del número de atributos que tiene y el tamaño de sus dominios. Si vemos que disponemos de muchos atributos para hacer una minidimensión, podemos hacer más de una. Si llevamos esto al extremo, podemos hacer una minidimensión para cada atributo de la Dimensión original.

3.2.4. Diseño de datos heterogéneos

Las instancias de algunas Dimensiones (como por ejemplo Productos) no son muy homogéneas. Los atributos que valen para un producto no tienen sentido para otro (por ejemplo, las neveras tienen litros de capacidad, mientras que los aparatos de vídeo disponen de sistema de grabación). Esto se traduce en la aparición de muchos valores nulos (que dificultan las consultas) y la imposibilidad de definir ciertas jerarquías de agregación porque no son válidas para todas las instancias.

Figura 37

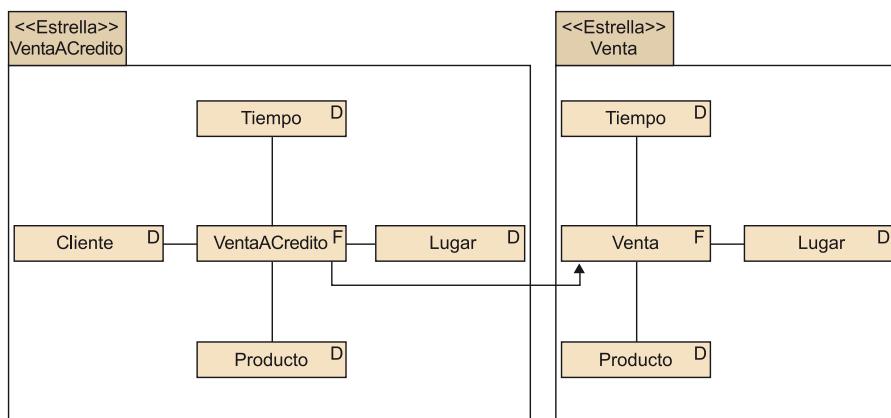


La manera de solucionar este problema consiste en especializar las Dimensiones (como podéis ver dibujado en la figura 37). Por un lado, tenemos la Dimensión general que tiene los Descriptores y jerarquía comunes a todas las instancias. Por el otro, definimos Dimensiones más pequeñas con Descriptores y jerarquías más específicas. Los usuarios podrán utilizar la Dimensión que más les interese en cada momento para consultar el Hecho.

Lo mismo puede suceder con los Hechos. Algunos acontecimientos tendrán más atributos o, incluso, más Dimensiones que otros. Del mismo modo que en el caso de las Dimensiones, lo que debemos hacer es especializar el Hecho.

Si solo cambia el número de atributos, podemos mantener una sola Estrella con más de un Hecho. Por el contrario, si según el tipo de hecho tenemos más o menos Dimensiones, es mejor definir una Estrella diferente.

Figura 38



Ejemplo de especialización de un Hecho

En la figura 38, podemos ver dos Estrellas en las cuales los Hechos respectivos están relacionados por una especialización. Mientras que si tenemos una venta a crédito podemos identificar al cliente, cuando el pago es en efectivo esta identificación no es posible. Por

lo tanto, la Dimensión Cliente no está presente en la Estrella para estudiar las ventas en general.

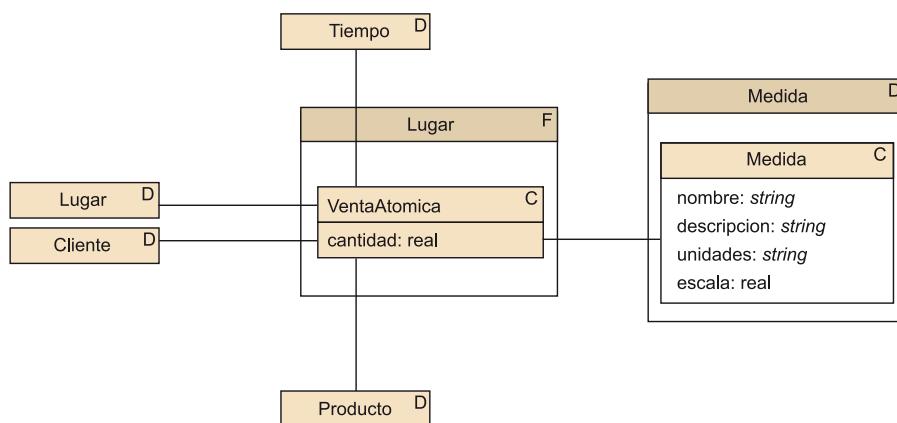
Cuando las instancias son heterogéneas, podemos especializar tanto los Hechos como las Dimensiones para evitar la aparición de Descriptores y Medidas inválidas, o facilitar la definición de jerarquías de agregación o Dimensiones específicas.

3.2.5. Hechos con una sola medida

Otro caso de heterogeneidad de las instancias se da cuando en cada acontecimiento tomamos diferentes medidas. Para especializar, es necesario diferenciar distintos tipos de Hechos y que las Medidas sean específicas de cada tipo concreto. Sin embargo, ¿qué podemos hacer cuando las medidas que tomamos dependen del acontecimiento que se da y no podemos definir tipos de acontecimientos?

Lo mejor en estos casos es definir una Dimensión Medida. El Hecho tendrá una sola Medida, que podemos denominar cantidad. Según a qué instancia de Medida asociamos una instancia del Hecho, cantidad tendrá un significado u otro. Además de ayudar a solucionar problemas de diferencias en las Medidas de cada acontecimiento, este tipo de diseño significa que las Medidas nos interesan de manera individual y no todas a la vez (con un diseño como este, no debería ser frecuente que consultáramos todas las mediciones que hicimos cuando se daba un acontecimiento).

Figura 39



Ejemplo de Hecho con una Medida

En la figura 39, podéis ver cómo tendríamos que modificar nuestra Estrella de ventas para que contuviese una sola Medida. Aprovechando que generamos una nueva Dimensión, podemos añadir Descriptores de las medidas, como por ejemplo las unidades, la escala, etc.

Como opción de diseño, podemos sustituir todas las Medidas de un Hecho por una sola, añadiendo una nueva Dimensión Medida.

4. Diseño lógico

Una vez hemos visto cómo se hace un diseño conceptual multidimensional, ahora veremos cómo se pasa al modelo lógico. En el mercado hay diferentes tipos de implementaciones multidimensionales: ROLAP, MOLAP o, incluso, O³LAP. Debido a la implantación actual de los sistemas relacionales, que cubren el mercado, la opción más común es la implementación ROLAP.

ROLAP, MOLAP, O³LAP

Los tres conceptos provienen del inglés. ROLAP es *relational OLAP*, MOLAP es *multidimensional OLAP* y O³LAP, *object-oriented OLAP*.

Una herramienta ROLAP no es más que una capa de software que recibe consultas multidimensionales, las traduce a SQL y las ejecuta sobre un SGBD relacional. Ahora veremos cómo se implementa un esquema multidimensional sobre las tablas de un SGBD de este tipo.

4.1. La Estrella (el caso básico)

Al igual que hacíamos en el diseño conceptual, para pasar al modelo lógico nos fijamos también en una Estrella cada vez. Además, en este apartado pensamos que el Hecho contiene una única Celda. Para implementar una Estrella, necesitamos una tabla para el Hecho (en el que cada fila representa una celda del espacio multidimensional) y una tabla más para cada una de las Dimensiones. Las jerarquías de agregación quedan implícitas en los valores de los atributos de las tablas de Dimensión. No los explicitamos con tablas diferentes.

Ejemplo de tabla de Dimensión con jerarquía de agregación implícita

La Dimensión Tiempo podría estar implementada con la relación siguiente:

Tiempo(RowID,diaMesAño,mesAño,semanaAño,año,festivo,laborable,traspaso,...)

Observad que esta relación no está normalizada. El atributo diaMesAño determina tanto mesAño como semanaAño. Además, cualquiera de estos dos atributos determina año. De este modo, sabemos a qué mes, semana y año pertenece cada uno de los días sin necesidad de definir la jerarquía de agregación. Las agrupaciones posibles de las filas quedan determinadas por los valores de los mismos atributos, en lugar de por una jerarquía explícita. Todas las filas con el mismo valor en el atributo año se agrupan para dar lugar a una instancia a Nivel Año.

Tabla de Dimensión

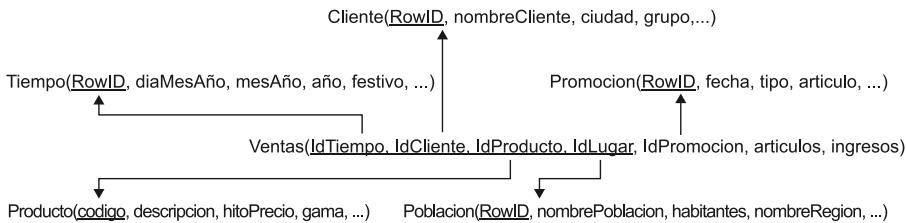
El término inglés para esta estructura de tablas y claves foráneas es *star join*.

La tabla de hecho estará relacionada por claves foráneas con las tablas de Dimensión.

Cada clave foránea apunta de la tabla del Hecho hacia una de las tablas de las Dimensiones. Junto a las claves foráneas, la tabla del Hecho contiene las Medidas, mientras que las tablas de Dimensión contienen los Descriptores. Si el Hecho no contiene Medidas, la tabla del Hecho solo contendrá las claves foráneas hacia las tablas de Dimensión.

Como clave primaria de la tabla del Hecho, tendremos los atributos correspondientes a una de las Bases de la Celda atómica. El resto de las Bases de la Celda darán lugar a claves alternativas. En cualquier caso, tanto la clave primaria como las alternativas serán subconjuntos del conjunto de claves foráneas que apuntan hacia las tablas de Dimensión.

Figura 40



Ejemplo de esquema relacional con forma de estrella

En el caso de las ventas y las promociones, obtendríamos el esquema relacional de la figura 40. Podemos ver que hay seis tablas: una para cada Dimensión (Tiempo, Producto, Poblacion, Promocion y Cliente) y una más para el Hecho (Ventas). Estas tablas están relacionadas por cinco claves foráneas, cada una de las cuales va de la tabla del Hecho a la clave primaria (el código de producto para la tabla Producto o el RowID para el resto de las tablas) de una de las tablas de las Dimensiones. Cuatro de estas claves foráneas forman la clave primaria de la tabla del Hecho. Si consideramos que {Promocion, Cliente, Dia, Poblacion} es una Base, entonces {IDPromocion, IDCiente, IDTiempo e IDLugar} sería clave alternativa de la tabla Ventas. Los atributos articulos e ingresos, las Medidas (que nunca son clave foránea), no forman parte de la clave primaria de Ventas.

Nota

Recordad que marcamos la clave primaria de una relación subrayando los atributos que la forman.

Cada Estrella da lugar a una tabla para el Hecho y una más para cada una de las Dimensiones. Estas tablas están vinculadas por claves foráneas que van de la tabla del Hecho a cada una de las tablas de Dimensión. La clave primaria de la tabla del Hecho es la concatenación de las claves foráneas correspondientes a una Base del Hecho.

Observad el tamaño de cada una de las tablas. La tabla del Hecho será de órdenes de magnitud mayor que cualquiera de las tablas de Dimensión. Ocupará más de un 95% del espacio utilizado por la Estrella. A primera vista, puede parecer extraño, pero una manera de reducir el tamaño de la tabla del Hecho consiste en definir sustitutos de la clave primaria en las tablas de Dimensión. Puesto que los sustitutos ocuparán menos espacio que los atributos identificadores de la misma tabla (por ejemplo, un DNI ocupa ocho caracteres, mientras que un RowID solo cuatro bytes), utilizándolos reducimos el tamaño de las columnas que forman la clave primaria de las tablas de Dimensión. Por lo tanto, también reducimos el tamaño de la clave primaria de la tabla del Hecho, porque sabemos que siempre está formada por claves foráneas que apuntan a las claves primarias de las tablas de Dimensión. Una pequeña ganancia en cada fila de la tabla del Hecho, que suele tener millones de filas, puede representar un gran ahorro de espacio.

Clave primaria de una tabla

Recordad que, como clave primaria de una tabla, no solo podemos tener atributos, sino también sustitutos (*surrogates*), conocidos también como *RowID*. Podéis ver el módulo "Reconsideración de los modelos conceptual y lógico" de la asignatura *Sistemas de gestión de bases de datos*.

El efecto colateral de la utilización de sustitutos de la clave primaria es que prevenimos posibles problemas ante cambios en los atributos identificadores en los sistemas operacionales. Recordad que queremos analizar largos perio-

dos de tiempo. Por lo tanto, es altamente probable que durante este periodo se produzcan recodificaciones de los identificadores de las promociones, de las poblaciones o de los clientes, con lo que también deberíamos recodificar todos los datos que tengamos en nuestra Estrella. Si utilizamos sustitutos para la clave primaria, ya no tenemos este problema, porque el identificador es absolutamente independiente del origen de los datos y cualquier cambio que se produzca no lo afectará.

Definimos sustitutos de la clave primaria en las tablas de Dimensión para reducir el tamaño de la tabla del Hecho. Además, también sirve para evitar problemas si se modifican los identificadores en los sistemas operacionales.

Además del tamaño, también hay diferencias en las operaciones que hacemos en cada tipo de tabla. Las tablas de Dimensión se crean con los datos en su interior y muy rara vez cambian. Solo de vez en cuando se añade una nueva fila o se cambia el valor de una que ya había (nunca se borran filas). En la tabla de Hechos, insertamos filas masivamente de manera regular y solo tiene modificaciones si hemos cometido un error durante la inserción (tampoco se borra nunca nada).

4.2. El copo de nieve

Ya hemos visto cómo se puede tratar el caso en el que solo tenemos una celda. Sin embargo, ¿qué hay que hacer cuando tenemos más de una dentro del mismo Hecho? Del mismo modo que antes, definimos una tabla de hecho para cada Celda, con las correspondientes claves foráneas hacia las tablas de Dimensión. Respecto a las tablas de Dimensión, ahora sí que hay que explicar los Niveles (normalizarlas), aunque solo parcialmente.

Daos cuenta que, si normalizamos totalmente una tabla de Dimensión, obtendremos una tabla diferente para cada uno de los Niveles. Hacerlo para todas las Dimensiones genera un esquema con forma de copo de nieve. La ganancia de espacio respecto del espacio total de la Estrella no resultaría significativa, porque las tablas de Dimensión son insignificantes respecto del volumen de datos que contiene la tabla del Hecho. Por el contrario, empeoraría el rendimiento de las consultas, porque cada vez que quisieramos operar con una Dimensión tendríamos que hacer la combinación (*join*) de las tablas de Dimensión correspondientes a los Niveles.

Estrella normalizada

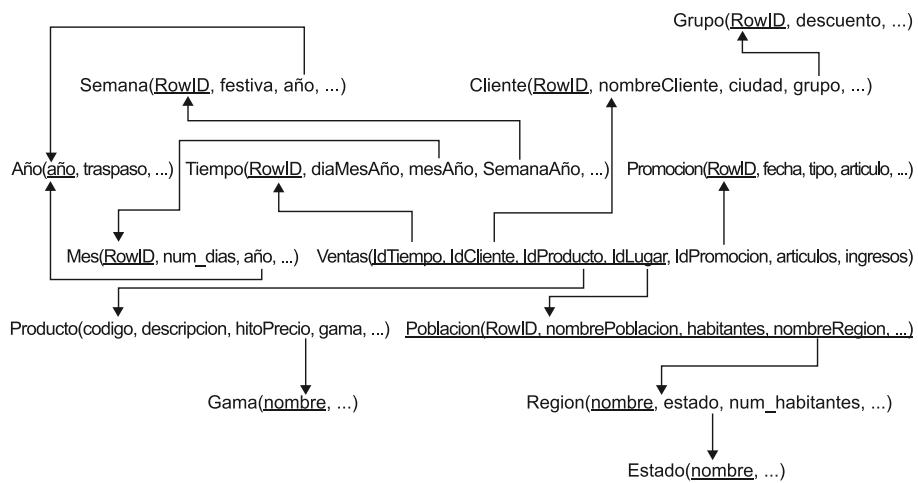
Una estrella completamente normalizada se conoce con el nombre de *copo de nieve* (*snowflake*), por su forma de estrella con ramificaciones, que recuerda la estructura de un copo de nieve.

La normalización es una herramienta del mundo transaccional que evita las redundancias y facilita la concurrencia en presencia de actualizaciones, pero aumenta el número de combinaciones necesarias para resolver algunas consultas. Tenemos que pensar que el objetivo de un esquema multidimensional es responder consultas de manera eficiente (como ya hemos dicho, no hay

actualizaciones), y resolver combinaciones no es nada rápido. Con las Dimensiones desnormalizadas, podemos mejorar el rendimiento de algunas consultas hasta un 30%.

En la figura siguiente, podéis ver cómo quedaría el esquema una vez normalizado. Podéis apreciar que, además de los problemas ya mencionados, ahora el esquema es más difícil de entender (sobre todo para usuarios no expertos). Desgraciadamente, además de ser más difícil de entender para los usuarios, también lo es para el optimizador de consultas. Algunos reconocen una estrella y utilizan técnicas específicas para la resolución de consultas, pero no hay ninguno que reconozca un copo de nieve.

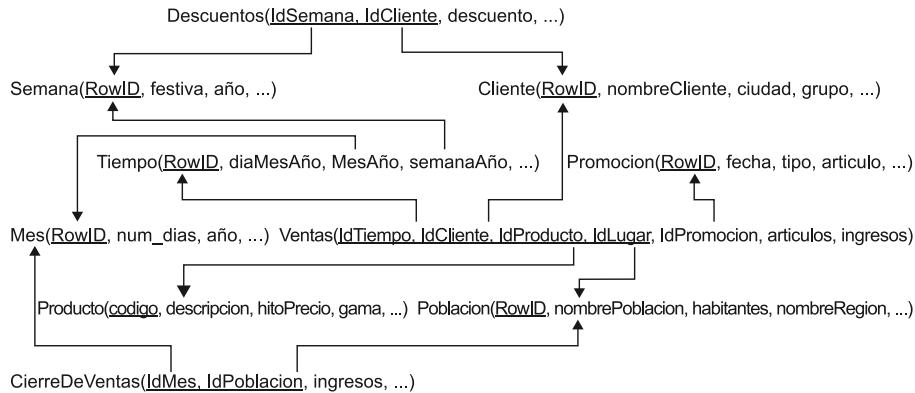
Figura 41



El copo de nieve es un error de diseño que genera una ganancia inapreciable de espacio y una gran pérdida de rendimiento. Para mejorar el rendimiento de las consultas, tenemos que evitar normalizar los esquemas, salvo casos extremos en los que el tamaño de la Dimensión sea comparable con el del Hecho.

Solo debemos definir tablas de Dimensión para los Niveles que tienen alguna Celda asociada. De este modo, las claves foráneas de la tabla del Hecho apuntarán a la granularidad correcta. No es posible que las Medidas correspondan a datos mensuales y la clave foránea de la asociación con la Dimensión Tiempo apunte a una tabla que contiene información de días.

Figura 42



Ejemplo de paso a relacional de un Hecho con múltiples Celdas

En la figura 42 podéis ver cómo quedaría un esquema relacional cuando tenemos diferentes Celdas en un mismo Hecho. En este caso, tenemos tres tablas de hecho (Descuentos, Ventas y CierreDeVentas). Cada una de estas apunta a sus tablas de Dimensión. Os tenéis que fijar especialmente en la normalización parcial que ha tenido la tabla Tiempo. Ahora hay dos tablas más (Semana y Mes) relacionadas con esta mediante claves foráneas. Observad que la información del año ya no está en la tabla Tiempo, sino en las tablas Semana y Mes (repetida en las dos). No se ha acabado de normalizar porque no hay ninguna Celda en Nivel Año.

Solo normalizaremos una Dimensión cuando el Hecho contenga Celdas diferentes y sea necesario hacerlo para relacionar la tabla de hecho correspondiente a cada Celda con la granularidad adecuada de la Dimensión.

4.3. Conformación: compartición de dimensiones

Generalmente, cada Estrella tiene sus Dimensiones. Sin embargo, es habitual que ciertas Dimensiones se utilicen en diferentes Estrellas (por ejemplo, la Dimensión Tiempo o Clientes). Hay que consensuar entre los analistas implicados cuál debe ser el contenido y formato de estas Dimensiones compartidas. Esto se denomina conformar las Dimensiones. De este modo, podremos utilizar en diferentes Estrellas tablas de Dimensión que tengan la misma forma, a pesar de que puedan estar implementadas sobre SGBD diferentes o, incluso, sobre máquinas distintas.

Conformar

Conformar significa hacer algo conforme a una norma.

Conformar las Dimensiones resulta extremadamente importante en el diseño de las Estrellas, para permitir navegar de una a otra y cambiar el tema objeto de análisis. Se tienen que conformar las Dimensiones que participen en Estrellas diferentes para hacer posible el *drill-across*.

Ventajas de tener las Dimensiones conformadas:

- Posibilita el *drill-across*.
- Ahorra trabajo de diseño y administración, porque la misma tabla se utiliza más de una vez. Si implementáramos todas las Estrellas dentro de un mismo SGBD, incluso podríamos tener solo una tabla para cada Dimensión, que sería compartida por tantas Estrellas como fuera necesario.
- Ayuda a consolidar la idea de una factoría de información empresarial, en vez de pequeños almacenes de datos departamentales aislados. Todos los usuarios dispondrán de los mismos Descriptores y jerarquías de agregación.

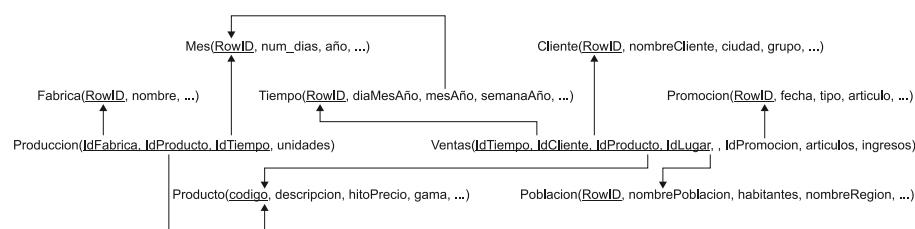
Conformación de Dimensiones

Un buen indicador para identificar cuándo dos Dimensiones que se tienen que conformar es que se denominen igual. Sin embargo, debéis tener cuidado con esto porque, por ejemplo, la Dimensión temporal de ventas no se puede conformar con la Dimensión temporal fiscal, la cual no viene marcada por las estaciones o días de fiesta, sino por los cierres de cuentas y fechas de declaración de impuestos.

Una Dimensión conformada es la que se utiliza en la implementación de más de una Estrella. A la hora de hacer consultas, tablas de Hecho que comparten tablas de Dimensión se pueden sustituir unas por otras (*drill-across*) sin modificar el resto de la consulta.

Compartir una Dimensión no significa que dos Estrellas deban utilizar exactamente la misma tabla de Dimensión. También se puede hacer si una Estrella utiliza una tabla que representa los datos de la Dimensión a un cierto grado de agregación y otra Estrella usa otra tabla de Dimensión que representa un grado de agregación superior. Si este es el caso, las dos tablas forman parte de la misma Dimensión.

Figura 43



Ejemplo de partición de tablas de Dimensión

En la figura 43 podemos ver cómo Ventas y Producción comparten Dimensiones. En primer lugar, la tabla de Dimensión Producto es apuntada por las dos tablas de Hecho. Por lo tanto, es preciso haber conformado la Dimensión Producto de las dos Estrellas. Además, no obstante, también tienen en común la Dimensión Tiempo. En este caso, cada Estrella la utiliza con una granularidad diferente. Producción la utiliza con grano-

laridad Mes, mientras que Ventas la usa con granularidad Día. Es preciso, pues, haber conformado también esta Dimensión.

Conformar las Dimensiones no implica que implementemos todas las Estrellas en el mismo SGBD, y ni siquiera lo tenemos que hacer en la misma máquina. Lo más importante es que las tablas que implementen la misma Dimensión en Estrellas diferentes posean la misma forma (los mismos atributos y dominios semánticos) para permitir pasar de una tabla de hecho a la otra. Si estas tablas se fusionan en una sola tabla o tenemos una tabla diferente en cada SGBD o en cada máquina, es un tema que no tenemos que decidir cuando se hace el diseño lógico.

4.4. Dimensiones degeneradas, de desechos y sombras

En algunos casos, podemos estar ante Dimensiones que no tienen atributos ni jerarquías de agregación, pero que nos son útiles simplemente para identificar las instancias del Hecho. Hablamos en estos casos de Dimensiones degeneradas, porque no dan lugar a ninguna tabla de Dimensión. Simplemente tenemos un atributo en la tabla de hecho, que forma parte de la clave, pero que no es clave foránea (no apunta a ninguna tabla de Dimensión).

Ejemplo de Dimensión degenerada

El ejemplo más habitual de Dimensión degenerada es el número que identifica el pedido. A pesar de no tener ningún atributo, nos resulta útil para identificar las ventas y agrupar todas las que se han hecho dentro del mismo pedido. Esta Dimensión Pedido no tiene ningún atributo, porque se los hemos sacado todos (fecha, cliente, etc.) para definir otras Dimensiones de análisis.

Las Dimensiones degeneradas suelen tener su origen en atributos identificadores de las bases de datos operacionales que coinciden con el gránulo elegido por la Estrella en cuestión. Es importante conservarlos porque permiten saber exactamente de dónde provienen los datos.

Una Dimensión degenerada es aquella que no da lugar a una tabla de Dimensión por falta de atributos, pero que sí se utiliza para identificar las instancias del Hecho.

Otro caso interesante se da al revés: cuando hay atributos que describen el Hecho, pero que no corresponden a ningún concepto en concreto que ayude a identificarlo. ¿Qué tenemos que hacer cuando tenemos un conjunto de atributos (frecuentemente booleanos) que provienen de los sistemas operacionales, independientes unos de otros? En este caso hay que crear una tabla de Dimensión auxiliar que contenga todos estos atributos, que no identificarán en ningún caso las instancias del Hecho, pero que utilizaremos para seleccionarlas y agregarlas.

Este tipo de tabla de Dimensión se denomina **Dimensión de desechos**. También relacionaremos este tipo de tablas en la tabla del Hecho con una clave foránea, pero en este caso no formará parte de la clave primaria.

Para contener todos aquellos atributos que provienen de los sistemas operacionales que no correspondan a ninguna Dimensión en concreto, crearemos una tabla de Dimensión especial, cuya clave no formará parte de la clave primaria del Hecho.

En las Dimensiones, también hay atributos que muy probablemente no nos servirán para seleccionar ni para definir jerarquías de agregación (como por ejemplo la dirección, el número de teléfono o simplemente un comentario a la Dimensión Cliente), pero que quizás querremos añadir a la consulta para generar el informe final justo antes de imprimirlo. Para no afectar al tiempo de respuesta, podemos poner todos estos atributos poco utilizados en otra tabla, que denominaremos **Dimensión sombra**.

Esta Dimensión sombra no estará relacionada directamente con la tabla de hecho, sino que tendremos en la Dimensión de verdad una clave foránea que la apuntará. De este modo, solo será preciso acceder a estos atributos (que habitualmente ocupan mucho espacio) para imprimir el informe final, y no afectarán al tiempo de respuesta durante la navegación por los datos.

Una Dimensión sombra es aquella tabla que contiene atributos descriptivos poco utilizados y que no relacionamos directamente con la tabla de hecho, sino con otra tabla de Dimensión.

4.5. Generalizaciones/especializaciones

Durante el diseño conceptual, cuando hay instancias heterogéneas, lo representamos con una especialización del Hecho o de la Dimensión, según sea el caso. Recordad las tres opciones que tenemos para implementar este tipo de vínculo entre clases:

- 1) Una tabla para la superclase y otra para cada subclase, todas con la misma clave primaria.
- 2) Una sola tabla que contenga tanto los atributos de la superclase como los de todas las subclases.
- 3) Solo una tabla para cada una de las subclases, repitiendo los atributos de la superclase en cada tabla.

En el diseño de bases de datos operacionales, la mejor opción es la primera (a pesar de que en algún caso muy especial, podría interesar más alguna de las otras dos). Sin embargo, en el caso del diseño multidimensional no es así.

Si la especialización es de una Dimensión (como por ejemplo, la de la figura 37), hay que considerar si las tablas estarán referenciadas por alguna tabla de hecho o no. Del mismo modo que no normalizamos para evitar tener que hacer combinaciones, ahora solo separaremos las subclases y superclase en tablas distintas cuando sea estrictamente necesario.

Si la tabla correspondiente a la superclase no es apuntada por una clave foránea desde ningún Hecho y las correspondientes a las subclases sí que lo son, lo mejor es la tercera opción, con la que nos ahorraremos combinaciones. Del mismo modo, si las tablas correspondientes a las subclases no son apuntadas por ninguna tabla de hecho, la mejor opción será la segunda. Por ejemplo, en el caso de la figura 37, si las Dimensiones *Neveras* y *Vídeos* no se utilizan en ninguna otra Estrella, sería mejor tener solo la tabla de Dimensión correspondiente a *Producto*, que en esta tabla también incluye los atributos específicos de neveras y vídeos.

Si la especialización es de una Dimensión, la implementaremos con una tabla de Dimensión para cada una de las clases que hay asociadas a algún Hecho.

Si lo que ha especializado es un Hecho (como tenéis dibujado en la figura 38), dada la cantidad de filas que tiene una tabla de hecho, la segunda opción queda descartada directamente por la cantidad de valores nulos que genera (con el consiguiente derroche de espacio y tiempo de consulta).

En cambio, a pesar de que pueda derrochar un poco de espacio, la tercera opción será la mejor, si cada analista está interesado en una sola subclase y quiere tratar juntos tanto los atributos propios de la subclase como los heredados de la superclase. Con esta opción, obtendríamos diferentes tablas de hecho (relativamente pequeñas) para cada analista, con la consiguiente ganancia de rendimiento. Hacer la unión de todas estas para obtener la superclase siempre es posible. No obstante, hay que decir que si normalmente no se quiere acceder al mismo tiempo a los atributos de la superclase y de las subclases, la mejor opción para implementar una especialización de un Hecho es la primera (una tabla de hecho para cada subclase y otra para la superclase).

Si la especialización es de un Hecho, tenemos que observar si se accede a los atributos de superclase y subclases al mismo tiempo o no. En caso de que la mayoría de las veces se acceda a estos al mismo tiempo, es mejor que solo tengamos tablas para las subclases. Cuando es más habitual acceder a los mismos por separado, entonces conviene tener los atributos de la superclase en una tabla distinta.

4.6. Estructuras temporales

Ya hemos dicho que la Dimensión temporal está prácticamente en todas las Estrellas. Los Hechos están asociados a la Dimensión temporal precisamente porque registra la evolución del negocio y sus cambios. Cada vez que se produce un acontecimiento, añadimos una nueva instancia del Hecho. Estas instancias corresponden a sucesos concretos (por ejemplo, ventas y movimientos en la cuenta corriente) o a estados (por ejemplo, stocks y saldos).

Visto de otro modo, hay dos posibilidades de registrar la evolución temporal: guardar las transacciones (diferenciales) o guardar sucesiones de fotografías. En el primer caso, el acontecimiento que hay que registrar estaría dado por una cierta acción en el mundo real de la que guardamos sus valores asociados. En el segundo caso, cada cierto tiempo registramos el estado de nuestro Hecho. Por un lado, podemos registrar cada venta concreta y, por el otro, podemos registrar en un momento dado cuánto se ha vendido.

Los dos mecanismos nos permiten mostrar la evolución del negocio, a pesar de que el que se basa en transacciones es mucho más detallado. A partir de este podemos construir las fotografías, pero no al revés. Aun así, los dos mecanismos son necesarios. Las transacciones nos dan el máximo grado de detalle, pero las fotografías nos permiten saber rápidamente el estado de la empresa en un momento dado.

Hay dos maneras de grabar los hechos. Lo podemos hacer mediante transacciones entre estados o por medio de fotografías del estado en un cierto momento.

Los Hechos siempre registran los cambios que se producen en el negocio. ¿Y las Dimensiones nunca registran ningún cambio? ¿No se modifican nunca? Dice una máxima budista que la única constante es que todo cambia. Por lo tanto, no nos tenemos que preguntar si las Dimensiones cambian, sino con qué frecuencia lo hacen.

Efectivamente, las Dimensiones cambian muy poco, pero también tienen algunos cambios (abriremos nuevas tiendas, cambiaremos el descuento que hacemos de manera habitual a un buen cliente, dejaremos de vender algún pro-

ducto, etc.). Si los cambios son realmente poco frecuentes, una posible solución consiste en definir una Estrella distinta cada vez que se produzca un cambio en una de sus Dimensiones. Otra solución cuando la Dimensión que cambia no es muy grande consiste en definir distintas versiones de la tabla de Dimensión y vincularlas todas al mismo cubo. En este caso, el usuario (o la interfaz que utiliza el cubo) debe saber qué tabla ha de utilizar para resolver la consulta en cada momento. La mejor solución y la más general, sin embargo, consiste en grabar los cambios dentro de la misma tabla de Dimensión afectada.

En los sistemas operacionales, registramos un cambio simplemente modificando el registro que toca. En un sistema de análisis, esto normalmente no se puede hacer de este modo. Imaginémonos que cambia el precio de un producto. Si sencillamente cambiamos el valor del atributo `precio` en la tabla de Dimensión, parecerá que todas las ventas se han hecho con este precio. No podremos ver la modificación que se produce en el volumen de ventas según cuál sea el precio del producto.

Cuando vemos que una Dimensión cambia, tenemos tres opciones:

- 1) Sobrescribir el valor antiguo con el nuevo (como se hace en los sistemas operacionales). Observad que si hacemos esto, renunciamos a estudiar cómo afecta este cambio al Hecho. Por otro lado, simplifica mucho el tratamiento y no utiliza más espacio del estrictamente necesario. Sin embargo, esta opción solo es aconsejable para los casos en que descubrimos que habíamos cometido un error en la introducción de datos y lo queremos enmendar sin dejar constancia del mismo. También lo es cuando el atributo en concreto no tiene ninguna incidencia en el análisis (por ejemplo, el nombre de una compañía). Si tiene incidencia en el análisis, porque define una característica utilizada en la selección o agrupación de instancias (como por ejemplo los habitantes de una población), esta opción no resulta nada apropiada (podríamos seleccionar hechos en una población con un cierto número de habitantes en un momento en el que todavía no los tenía).
- 2) Crear una nueva fila en la tabla de Dimensión (con su *RowID*) que será referenciada desde ahora por todas las instancias de hecho que añadamos. Esta es la solución más habitual. Observad que implica la utilización de un sustituto de la clave primaria en la tabla de Dimensión. ¿Cómo podríamos identificar la segunda fila dentro de la tabla? Si utilizáramos el DNI como clave primaria, no podríamos registrar de este modo los cambios en los datos de una persona, porque las dos filas deberían tener el mismo DNI.

Otra razón para utilizar sustitutos de la clave primaria en las tablas de Dimensión consiste en registrar los cambios sin tener problemas de identificación.

Observad que con esta solución, lo que hacemos realmente es crear una nueva instancia de la Dimensión. En cierto modo decimos que hay dos personas distintas, una antes del cambio y otra después. Para hacer consultas, no hay que ser consciente del cambio. La misma condición ya indicará cuál de las dos filas se elige y, por lo tanto, en qué hechos estamos interesados.

Un problema de esta opción es que no será evidente que dos filas hacen referencia a la misma instancia en momentos distintos. Esto se puede solucionar manteniendo (además del *RowID*) un atributo identificador (como por ejemplo el DNI) y añadir un atributo más que sea el número de versión de la instancia. Tendremos tantas versiones de una instancia como cambios haya en sus atributos.

3) Tener diferentes atributos en la tabla de Dimensión para registrar el valor antiguo y el nuevo. Es decir, para cada atributo que pueda tener un cambio, ahora utilizamos dos atributos. Esta solución, como ya os podéis imaginar, es muy limitada y útil solo en casos muy concretos. En primer lugar, es preciso que el cambio sea general en todas las filas de la tabla. No obstante, además, solo funciona si hay un único cambio. Si hubiera dos, necesitaríamos tres atributos; si hubiera tres, cuatro atributos, etc. Una implementación como esta puede resultar útil, por ejemplo, para cambiar los nombres de las poblaciones del español al catalán (o del español al euskera). Las poblaciones son exactamente las mismas (no generamos nuevos *RowID*). Simplemente, queremos utilizar la nueva denominación y no queremos perder la denominación española. Con esta opción, en algunos casos, además del atributo con el valor antiguo también es preciso otro atributo con la fecha del cambio.

Básicamente, encontramos tres posibilidades para registrar los cambios en las tablas de Dimensión:

- Modificar el valor directamente.
- Crear una nueva fila.
- Crear una nueva columna.

Recordad las minidimensiones de las que hablábamos en el apartado de diseño conceptual. Observad que definíamos perfiles de usuario en lugar de propiamente usuarios. En este tipo de Dimensión, no reflejaremos los cambios de ninguna de estas tres maneras. Si cambia uno de los atributos del usuario, simplemente asociaremos sus acontecimientos con un perfil distinto. Sin em-

bargo, la tabla de Dimensión no tendrá ninguna modificación a menos que este cambio genere un nuevo perfil de usuario, que no había antes. Si este es el caso, simplemente lo añadiremos ahora a la tabla.

Estas minidimensiones también ayudan a gestionar los cambios en las Dimensiones. Normalmente, tenemos atributos que cambian con más frecuencia que otros. La edad o los ingresos anuales tienen una frecuencia de cambio mucho más alta que la población de residencia, el nombre o el sexo. Para no tener que añadir una nueva fila en la tabla Cliente cada vez que cambien estos atributos, los podemos meter dentro de una minidimensión demográfica, en la que ya hemos dicho que no hay que añadir nuevas filas cada vez que se produce un cambio, sino simplemente relacionar cada instancia del Hecho con el perfil correspondiente.

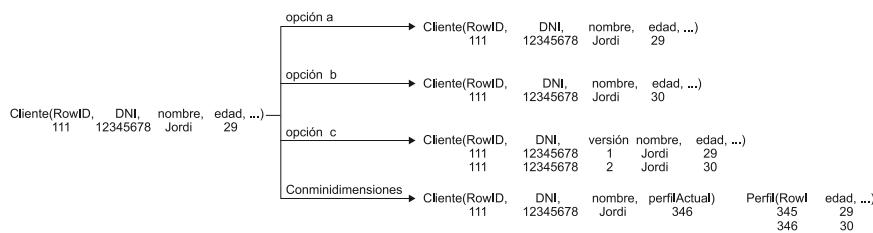
Los cambios en las minidimensiones son más fáciles de gestionar que en las Dimensiones normales, porque no hay que modificarlas. En algunos casos, crearlas puede ser una buena opción para registrar los cambios.

Generalmente, podéis distribuir los atributos de cualquier Dimensión según la frecuencia de cambio, para facilitar su gestión. De este modo, tendríais una tabla de Dimensión que no cambia casi nunca y otra que cambia cada cierto tiempo.

Ejemplo de gestión de los cambios en las Dimensiones

En la figura siguiente, podéis ver las cuatro maneras de reflejar el cambio de edad de Jorge de los 29 a los 30 años. En el caso de utilizar minidimensiones, la tabla de hecho tendría una clave foránea hacia Perfil, para dejar constancia del perfil que tenía Jorge en el momento en que se produjo el acontecimiento.

Figura 44



5. Consultas con SQL'99

Ahora que ya hemos visto cómo se implementan las estrellas en un SGBD relacional, veamos cómo las podemos consultar con SQL estándar. Además de ver la estructura básica de una consulta, también observaremos las cláusulas especiales que incorpora la especificación del estándar de 1999 (SQL'99).

5.1. Estructura básica de la consulta

Lo que tenemos en un esquema con forma de estrella es una tabla de Hecho y una tabla para cada una de las Dimensiones. Para hacer una consulta, tenemos que poner todas estas tablas en la cláusula FROM. En WHERE relacionaremos cada tabla de Dimensión con la tabla de hecho utilizando la clave foránea correspondiente (nunca vincularemos Dimensiones entre sí) y añadiremos las condiciones que queremos sobre los atributos de las tablas de Dimensión. En SELECT pondremos las Medidas de la tabla de hecho que queramos visualizar con la correspondiente operación de agregación. Finalmente, pondremos la cláusula GROUP BY con los identificadores de los Niveles en los que queremos ver los datos en cada Dimensión. Siempre añadiremos los atributos que aparezcan en GROUP BY en SELECT para distinguir las filas. Habitualmente, también se añade la cláusula ORDER BY con todos los atributos de las tablas de Dimensión que tenemos en SELECT, para obtener el resultado ordenado y facilitar su visualización.

```
SELECT Nivel1, ..., Niveln, Operacion(Medida), ...
FROM Hecho, Dimension1, ..., Dimensionn
WHERE Hecho=Dimension1 AND ... AND Hecho=Dimensionn
AND Descriptor1=valor AND ... AND Descriptorm=valor
GROUP BY Nivel1, ..., Niveln
ORDER BY Nivel1, ..., Niveln
```

Cada una de las operaciones del modelo multidimensional que hemos visto en el apartado correspondiente tiene una relación directa con esta estructura de consulta.

- Selección: para seleccionar unos puntos u otros de nuestro espacio *n*-dimensional, lo que tenemos que hacer es añadir o quitar condiciones sobre los Descriptores en la cláusula WHERE.
- Proyección: quitándolas de SELECT, dejamos de ver las Medidas que no nos interesan.

- *Roll-up*: para aumentar o disminuir el nivel de detalle con el que vemos los datos, solo hay que agrupar según el atributo que identifica el Nivel u otro de la Dimensión que corresponde. Hay que poner los atributos correspondientes a GROUP BY. En caso de querer hacer *roll-up* hasta el Nivel All, lo que hace falta es quitar todos los atributos de la Dimensión.
- *Drill-across*: si queremos cambiar el tema de análisis, solo hay que sustituir la tabla de hecho en la cláusula FROM (recordad que para poder hacerlo es preciso que los dos Hechos compartan Dimensiones). Las Dimensiones que no hay en el nuevo Hecho desaparecerán.
- CambioBase: finalmente, si lo que queremos es ver los mismos datos ordenados de manera distinta, solo hay que modificar los atributos de SELECT y cambiar el orden de los atributos a ORDER BY.

Ejemplo de secuencia de operaciones sobre una consulta SQL

Dada la consulta:

```
SELECT d1.nombre_articulo, d2.nombre_fabrica, d3.mesAño,
       SUM(hecho.unidades)
  FROM Produccion hecho, Producto d1, Fabrica d2, Tiempo d3
 WHERE hecho.IDProducto=d1.id
       AND hecho.IDFabrica=d2.id
       AND hecho.IDTiempo=d3.id
       AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
       AND d2.num_trabajadores>100
       AND d3.mesAño IN ('Enero2002', 'Febrero2002')
 GROUP BY d1.nombre_articulo, d2.nombre_fabrica, d3.mesAño
 ORDER BY d1.nombre_articulo, d2.nombre_fabrica, d3.mesAño;
```

A := roll-up_{Fábrica::All}("Unidades producidas por Producto, Fábrica y Mes")

```
SELECT d1.nombre_articulo, 'All', d3.mesAño, SUM(hecho.unidades)
  FROM Produccion hecho, Producto d1, Fabrica d2, Tiempo d3
 WHERE hecho.IDProducto=d1.id
       AND hecho.IDFabrica=d2.id
       AND hecho.IDTiempo=d3.id
       AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
       AND d2.num_trabajadores>100
       AND d3.mesAño IN ('Enero2002', 'Febrero2002')
 GROUP BY d1.nombre_articulo, d3.mesAño
 ORDER BY d1.nombre_articulo, d3.mesAño;
```

B := cambioBase_{Producto×Tiempo}(A)

```
SELECT d1.nombre_articulo, d3.mesAño, SUM(hecho.unidades)
  FROM Produccion hecho, Producto d1, Fabrica d2, Tiempo d3
 WHERE hecho.IDProducto=d1.id AND hecho.IDTiempo=d3.id
       AND hecho.IDFabrica=d2.id
       AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
       AND d2.num_trabajadores>100
       AND d3.mesAño IN ('Enero2002', 'Febrero2002')
 GROUP BY d1.nombre_articulo, d3.mesAño
 ORDER BY d1.nombre_articulo, d3.mesAño;
```

C := drill-across_{Ventas}(B)

```
SELECT d1.nombre_articulo, d3.mesAño, SUM(hecho.articulos),
       SUM(hecho.ingresos)
  FROM Ventas hecho, Producto d1, Tiempo d3
 WHERE hecho.IDProducto=d1.id
```

```

    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d3.mesAño
ORDER BY d1.nombre_articulo, d3.mesAño;

```

$D := \text{proyección}_{\text{artículos}}(C)$

```

SELECT d1.nombre_articulo, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d3.mesAño
ORDER BY d1.nombre_articulo, d3.mesAño;

```

$E := \text{cambioBase}_{\text{Artículos} \times \text{Lugar} \times \text{Tiempo}}(D)$

```

SELECT d1.nombre_articulo, 'All', d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d3.mesAño
ORDER BY d1.nombre_articulo, d3.mesAño;

```

$F := \text{drill-down}_{\text{Lugar}::\text{Región}}(E)$

```

SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d2.region, d3.mesAño
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;

```

$R := \text{selección}_{\text{Región.nombre}=\text{"Cataluña}}(F)$

```

SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d2.region, d3.mesAño
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;

```

5.2. GROUPING SETS

Lo más habitual es que los usuarios no solo quieran ver unos ciertos datos, sino también los totales. Es decir, el resultado de una consulta sería una tabla como esta:

Ventas	Cataluña		
	Enero 2002	Febrero 2002	Total
Bolígrafos	275.827 (a)	290.918 (a)	566.745 (c)
Gomas	784.172 (a)	918.012 (a)	1.702.184 (c)

Ventas	Cataluña		
	Enero 2002	Febrero 2002	Total
Total	105.999 (b)	1.208.930 (b)	2.268.929 (d)

Esta tabla no es propiamente un cubo, porque mezcla celdas de cuatro granularidades distintas (las a, las b, las c y la d). Aun así, la podemos entender como la unión de cuatro cubos. Del mismo modo que podemos definir una función por trozos, también podemos definir la tabla por trozos uniendo cubos.

Ventas (a) \oplus Roll-up_{Tiempo::All}(Ventas) (b) \oplus Roll-up_{Artículos::All}(Ventas) (c) \oplus Roll-up_{Artículos::All, Tiempo::All}(Ventas) (d)

Observad que la consulta SQL que hemos visto antes solo nos muestra las cuatro celdas (a). Para conseguir las otras cinco celdas sería necesario cuatro consultas, como se hace en la consulta siguiente:

(a)

```
SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos','Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d2.region, d3.mesAño
UNION
```

(b)

```
SELECT d1.nombre_articulo, d2.region, 'Total', SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos','Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d1.nombre_articulo, d2.region
UNION
```

(c)

```
SELECT 'Total', d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos','Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002','Febrero2002')
GROUP BY d2.region, d3.mesAño
UNION
```

(d)

```
SELECT 'Total', d2.region, 'Total', SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
```

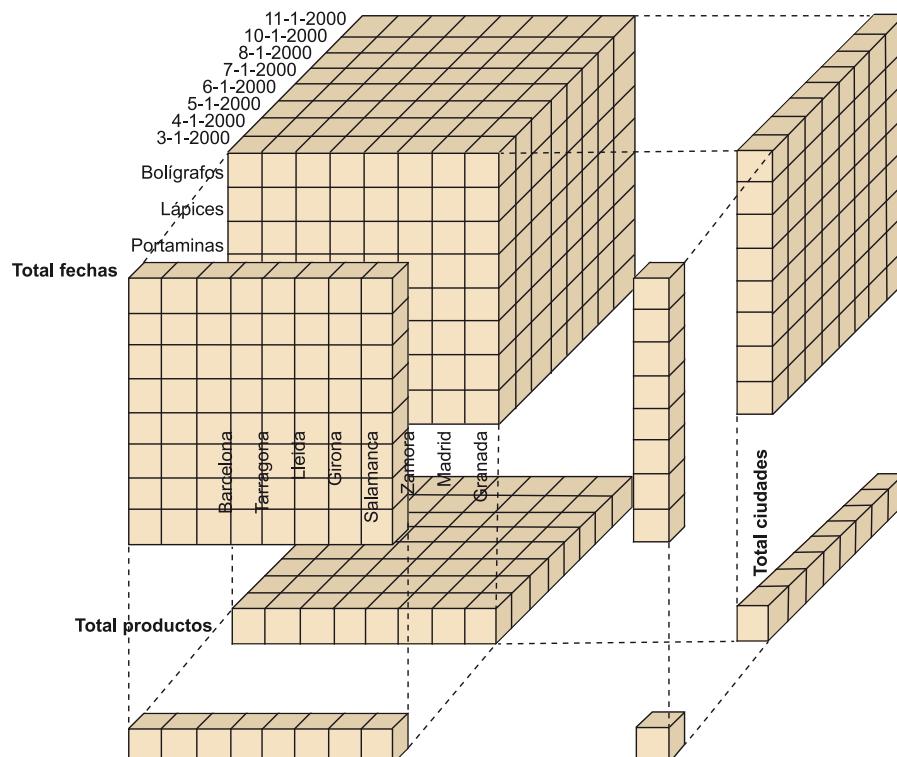
```

WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002', 'Febrero2002')
GROUP BY d2.region
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;

```

Si en lugar de calcular el total para dos Dimensiones lo quisiéramos hacer para tres, necesitaríamos siete uniones. La figura siguiente os muestra esquemáticamente cuáles son los ocho cubos que habría que unir (ventas por día, producto y ciudad; por día y producto; por día y ciudad; por producto y ciudad; por día; por producto; por ciudad, y el total de ventas).

Figura 45



El número de uniones que hay que hacer para calcular los totales crece de manera exponencial respecto al número de Dimensiones que tengamos. Por suerte, el estándar SQL'99 ya nos ofrece otra sintaxis para abreviar todas estas uniones. Como podéis ver en esta consulta, solo hay que poner en GROUP BY las palabras clave 'GROUPING SETS' y, entre paréntesis, la lista de agrupaciones que se quiere hacer.

```

SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProducto=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002', 'Febrero2002')
GROUP BY GROUPING SETS ((d1.nombre_articulo, d2.region, d3.mesAño),
                        (d1.nombre_articulo, d2.region),
                        (d2.region, d3.mesAño),
                        ());

```

```
(d2.region))
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;
```

Para calcular diferentes agrupaciones de la misma consulta sin tener que explicitar las uniones, el estándar SQL'99 nos ofrece las palabras reservadas GROUPING SETS dentro de la cláusula GROUP BY.

El resultado de la consulta anterior sería este:

Nombre_articulo	Región	MesAño	Artículos
Bolígrafos (a)	Cataluña (a)	Enero02 (a)	275827 (a)
Bolígrafos (a)	Cataluña (a)	Febrero02 (a)	290918 (a)
Bolígrafos (c)	Cataluña (c)	NULL (c)	566745 (c)
Gomas (a)	Cataluña (a)	Enero02 (a)	784172 (a)
Gomas (a)	Cataluña (a)	Febrero02 (a)	918012 (a)
Gomas (c)	Cataluña (c)	NULL (c)	1702184 (c)
NULL (b)	Cataluña (b)	Enero02 (b)	1059999 (b)
NULL (b)	Cataluña (b)	Febrero02 (b)	1208930 (b)
NULL (d)	Cataluña (d)	NULL (d)	2268929 (d)

A parte del problema evidente de la presentación, que tiene que arreglar la misma interfaz gráfica, observad los valores nulos. ¿Qué significan, 'desconocido' o 'inaplicable'? Pues ninguna de las dos opciones. Este es un tercer significado de los valores nulos que se ha definido en el SQL'99. En esta tabla, quieren decir 'total'.

El valor nulo no solo significa 'desconocido' o 'inaplicable'. En el contexto de los GROUPING SETS, también puede querer decir 'total'.

Esto plantea un nuevo problema. ¿Cómo sabemos si un valor nulo significa que no conocemos el valor del atributo o que la fila es el resultado de aplicar una operación de agregación? Para contestar a esta pregunta, el estándar define la función GROUPING.

La función GROUPING tiene como parámetro un atributo. Retorna "1" si ese atributo toma el valor 'total'. En cualquier otro caso, retorna "0". Con esta función, podemos reescribir la consulta anterior de modo que aparezca la palabra total en lugar de los valores nulos siempre que convenga (dados que d2.region aparece en las cuatro agrupaciones, sabemos que no puede tomar el valor 'total').

```

SELECT
    CASE WHEN GROUPING(d1.nombre_articulo)=1
        THEN 'TotalDeEneroYFebrero'
        ELSE d1.nombre_articulo,
    d2.region,
    CASE WHEN GROUPING(d3.mesAño)=1
        THEN 'TotalDeBolígrafosYGomas'
        ELSE d3.mesAño,
    SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProdcte=d1.id
    AND hecho.IDLugar=d2.id
    AND hecho.IDTiempo=d3.id
    AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
    AND d2.region='Cataluña'
    AND d3.mesAño IN ('Enero2002', 'Febrero2002')
GROUP BY GROUPING SETS ((d1.nombre_articulo, d2.region, d3.mesAño),
                        (d1.nombre_articulo, d2.region),
                        (d2.region, d3.mesAño),
                        (d2.region))
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;

```

Ahora, el resultado de la consulta sería este:

Nombre_articulo	Región	mesAño	Artículos
Bolígrafos (a)	Cataluña (a)	Enero02 (a)	275827 (a)
Bolígrafos (a)	Cataluña (a)	Febrero02 (a)	290918 (a)
Bolígrafos (c)	Cataluña (c)	TotalDeEneroYFebrero (c)	566745 (c)
Gomas (a)	Cataluña (a)	Enero02 (a)	784172 (a)
Gomas (a)	Cataluña (a)	Febrero02 (a)	918012 (a)
Gomas (c)	Cataluña (c)	TotalDeEneroYFebrero (c)	1702184 (c)
TotalDeBolígrafosYGomas (b)	Cataluña (b)	Enero02 (b)	1059999 (b)
TotalDeBolígrafosYGomas (b)	Cataluña (b)	Febrero02 (b)	1208930 (b)
TotalDeBolígrafosYGomas (d)	Cataluña (d)	TotalDeEneroYFebrero (d)	2268929 (d)

Podéis distinguir el significado de un valor nulo utilizando la función GROUPING.

5.2.1. ROLLUP

Como ya hemos dicho, el número de totales crece de manera exponencial respecto al número de Dimensiones. Por lo tanto, aunque no tengamos que escribir toda la consulta, escribir solo todas las combinaciones de atributos en GROUPING SETS ya puede llegar a resultar demasiado complejo. Para facilitar todavía más este tipo de consulta, el estándar también define la palabra reservada ROLLUP. Con un cierto conjunto de atributos, calcula todas las agrupa-

El orden de los atributos

El orden de los atributos de agrupación dentro de ROLLUP sí afecta al resultado de la consulta. El orden de los atributos dentro de ORDER BY no afecta al resultado de la misma.

ciones que resultan de ir ignorando los atributos uno por uno, de derecha a izquierda. Veamos cómo se incorporaría a la consulta anterior (prestad atención al orden de los atributos en GROUP BY y ORDER BY).

```
SELECT d1.nombre_articulo, d2.region, d3.mesAño,
       SUM(hecho.articulos)
  FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
 WHERE hecho.IDProdctue=d1.id
   AND hecho.IDLugar=d2.id
   AND hecho.IDTiempo=d3.id
   AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
   AND d2.region='Cataluña'
   AND d3.mesAño IN ('Enero2002', 'Febrero2002')
 GROUP BY ROLLUP (d2.region, d1.nombre_articulo, d3.mesAño);
 ORDER BY d2.region, d3.mesAño, d1.nombre_articulo;
```

El resultado de esta consulta sería el siguiente:

nombre_articulo	región	mesAño	artículos
Bolígrafos (a)	Cataluña (a)	Enero02 (a)	275.827 (a)
Gomas (a)	Cataluña (a)	Enero02 (a)	784.172 (a)
Bolígrafos (a)	Cataluña (a)	Febrero02 (a)	290.918 (a)
Gomas (a)	Cataluña (a)	Febrero02 (a)	918.012 (a)
Bolígrafos (c)	Cataluña (c)	NULL (c)	566.745 (c)
Gomas (c)	Cataluña (c)	NULL (c)	1.702.184 (c)
NULL (d)	Cataluña (d)	NULL (d)	2.268.929 (d)
NULL (d)	NULL (d)	NULL (d)	2.268.929 (d)

Las primeras cuatro filas corresponden a "GROUP BY d2.region, d1.nombre_articulo, d3.mesAño"; las dos siguientes a "GROUP BY d2.region, d1.nombre_articulo"; la siguiente a "GROUP BY d2.region"; y la última a "GROUP BY ()". Dado que solo disponemos de un valor para d2.region, las dos últimas filas tienen el mismo valor. Podemos evitar que salga la última fijando este atributo e indicando que se utilice en todas las agrupaciones.

Nueva sintaxis

```
SELECT COUNT(*)
  FROM tabla;
```

Con SQL'99, ahora también se puede escribir:

```
SELECT COUNT(*)
  FROM tabla
 GROUP BY ();
```

```
SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
  FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
 WHERE hecho.IDProdctue=d1.id
   AND hecho.IDLugar=d2.id
   AND hecho.IDTiempo=d3.id
   AND d1.nombre_articulo IN ('Bolígrafos', 'Gomas')
   AND d2.region='Cataluña'
   AND d3.mesAño IN ('Enero2002', 'Febrero2002')
 GROUP BY d2.region, ROLLUP (d1.nombre_articulo, d3.mesAño);
 ORDER BY d2.region, d3.mesAño, d1.nombre_articulo;
```

El resultado de esta consulta es el mismo de antes, menos la última fila. Con esto, podemos reescribir la consulta original de cuatro agrupaciones de la manera siguiente:

```
SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
  FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
```

```
WHERE hecho.IDProdcute=d1.id  
      AND hecho.IDLugar=d2.id  
      AND hecho.IDTiempo=d3.id  
      AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')  
      AND d2.region='Cataluña'  
      AND d3.mesAño IN ('Enero2002', 'Febrero2002')  
GROUP BY GROUPING SETS  
      ((d2.region, ROLL-UP (d1.nombre_articulo, d3.mesAño)),  
       (d2.region, d3.mesAño))  
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;  
  
GROUP BY ROLL-UP (a1,...,an)
```

equivale a:

GROUP BY

```

        (a1,...,an-1),
        ...
        (a1),
        ())

```

5.2.2. CUBE

Con ROLLUP ya no hace falta que escribamos todas las combinaciones de atributos que nos interesen, pero todavía tenemos que escribir algunas de las mismas. Las podemos conseguir todas directamente si utilizamos la palabra reservada *CUBE* en vez de *ROLLUP*. Con la consulta siguiente, obtenemos las ocho celdas que queríamos originalmente:

```

SELECT d1.nombre_articulo, d2.region, d3.mesAño, SUM(hecho.articulos)
FROM Ventas hecho, Producto d1, Lugar d2, Tiempo d3
WHERE hecho.IDProduct=d1.id
      AND hecho.IDLugar=d2.id
      AND hecho.IDTiempo=d3.id
      AND d1.nombre_articulo IN ('Boligrafos', 'Gomas')
      AND d2.region='Cataluña'
      AND d3.mesAño IN ('Enero2002', 'Febrero2002')
GROUP BY d2.region, CUBE (d1.nombre_articulo, d3.mesAño);
ORDER BY d1.nombre_articulo, d2.region, d3.mesAño;

```

GROUP BY CUBE (a,b,c)

equivale a:

```
GROUP BY GROUPING SETS ((a,b,c),  
                         (a,b),  
                         (a,c),  
                         (b,c),  
                         (a),  
                         (b),  
                         (c),  
                         () );
```

También podemos combinar CUBE y ROLLUP para obtener los totales que nos interesen.

GROUP BY CUBE(a, b), ROLLUP(c, d)

es equivalente a:

```
GROUP BY GROUPING SETS ((a,b,c,d),  
                         (a,b,c),  
                         (a,b),  
                         (a,c,d),  
                         (a,c),  
                         (a)).
```

```
(b, c, d),  
(b, c),  
(b),  
(c, d),  
(c),  
( ) )
```

El estándar permite combinar CUBE, ROLLUP y GROUPING SETS de cualquier manera para obtener el resultado deseado.

Utilizar CUBE, ROLLUP y GROUPING SETS, además de facilitar la escritura de las consultas, también mejora el rendimiento del sistema porque da información extra al optimizador de consultas sobre lo que se pretende hacer.

6. Diseño físico

Una vez ya tenemos las relaciones que componen nuestra base de datos y también sabemos qué tipo de consultas queremos ejecutar, solo queda hacer el diseño físico teniendo en mente que hay que conseguir un buen tiempo de respuesta a las consultas.

6.1. Plan y técnicas básicas de acceso

Antes de ver qué herramientas específicas tenemos desde el punto de vista físico para mejorar el rendimiento del sistema, pensad cómo sería el plan de acceso de una consulta multidimensional (este plan puede tener pequeñas variaciones según la SGBD):

- 1) Evaluar las condiciones sobre cada una de las Dimensiones para obtener un conjunto de identificadores.
- 2) Combinar (hacer el producto cartesiano) los identificadores de todas las Dimensiones para obtener los identificadores del Hecho que nos interesan.
- 3) Buscar el Hecho para obtener los valores (mediciones) que queríamos.
- 4) Ordenar los resultados.
- 5) Agrupar y operar mediciones.

SGBD

Hay que asegurarse de que el SGBD reconoce el esquema con forma de estrella y utiliza este plan de acceso.

En el segundo paso del plan de acceso que acabamos de ver, se hace el producto cartesiano de todos los identificadores seleccionados en las Dimensiones. A pesar de que esta operación es muy costosa, no lo es tanto como hacer de manera secuencial la combinación de la tabla de Hecho (sin duda, la más grande de todas) con cada una de las tablas de Dimensión.

La pregunta que nos podemos hacer ahora es cómo se puede abaratar todavía más este segundo paso. ¿Podemos evitar el producto cartesiano? La respuesta es que sí, mediante índices de combinación (*join indices*). Un índice de combinación es aquel definido sobre una clave foránea, de modo que tiene los valores de una tabla y apunta a la otra. Como podéis ver en la figura siguiente, en cierto modo, el índice contiene el resultado de la combinación precalculado.

Figura 46

Cliente (RowID, DNI, nombre, edad...)				Índice de combinación	Ventas (clientelD...)
111	12345678	Jordi	29	111 → 1,2	111
112	87654321	Manel	40	112 → 3,4	111 112 112

Cuando definimos una clave primaria en una tabla, el SGBD define un índice B⁺-árbol sobre los atributos correspondientes. Observad que en una tabla de hecho, la clave primaria está formada por claves foráneas hacia las tablas de Dimensión. Por lo tanto, realmente hablamos de un índice de combinación. Con los identificadores de una Dimensión podemos recorrer el índice hasta tener la rama o ramas que nos interesan: no será necesario hacer el producto cartesiano de todas las Dimensiones.

Figura 47

Índice de combinación	Ventas (ClientelD, TiempolD...)
111 → 222 → 1	111 222
111 → 223 → 2	111 223
112 → 333 → 3	112 333
112 → 334 → 4	112 334

El problema que tiene este tipo de índice es que es preciso que las condiciones de la consulta afecten a las Dimensiones correspondientes a los primeros atributos de la clave primaria. Recordad que en un índice B⁺-árbol el orden de los atributos es relevante. No es lo mismo construir un índice sobre la fecha y los clientes, que sobre los clientes y la fecha. Mientras que para utilizar el primero es preciso haber fijado la fecha, para usar el segundo se tiene que haber fijado al cliente. Siempre entramos en el índice por el valor correspondiente al primer atributo y comprobamos los valores del resto de los atributos de manera secuencial. Por lo tanto, para utilizar un índice B⁺-árbol, es preciso que el usuario haya fijado el valor o los valores del primer atributo. Dado que muchas consultas multidimensionales se hacen restringiendo la fecha, esta sería una buena elección para el primer atributo de un índice de combinación.

En una tabla, además del índice de la clave primaria, podemos definir tantos índices como queramos, pero definir uno para cada combinación de Dimensiones que pueda llegar a pedir el usuario es posible que signifique sobrecargar el sistema con la gestión de índices inútiles.

Definición de los índices

Definir mal los índices puede significar que una consulta tarda horas en ejecutarse.

Los índices B⁺-árbol son especialmente útiles para hacer consultas simples (sin agrupaciones, ni agregaciones, ni muchas combinaciones), y funcionan mejor cuanto mayor es la selectividad del atributo (cuanto menos valores repetidos tiene). El principal problema en este caso es que los atributos de las consultas multidimensionales no suelen ser muy selectivos. Además, para tablas muy

grandes, el índice B⁺-árbol puede ocupar demasiado espacio. Si la tabla de Hecho contiene menos de cinco Medidas, un índice B⁺-árbol puede ocupar un 80% del tamaño de la tabla.

Los índices B⁺-árbol pueden ser útiles para resolver algunas consultas multidimensionales, pero no es suficiente con esto.

Observad, también, que definir un índice como agrupado (*cluster*) puede resultar muy provechoso y no muy costoso si el primer atributo es la fecha. Dado que las inserciones se hacen de manera masiva y de fecha en fecha, siempre irán a parar al final de la tabla. Si la tabla está ordenada por fecha, los datos que insertamos ayer tienen que estar antes que los que insertamos hoy, que han de estar antes que los que insertaremos mañana, etc. De este modo, la tabla queda ordenada sin ningún coste adicional.

Índice agrupado

Recordad que un índice agrupado es aquel que no solo mantiene ordenado el índice, sino también los datos en el interior de la tabla.

6.2. Índices de mapas de bits

La mejor opción (aunque no disponible en todos los SGBD) para indexar tablas de Hecho son los índices de mapa de bits (*bitmap*). Un índice de mapa de bits es una matriz booleana de tantas columnas como valores tenga el atributo utilizado para la indexación y tantas filas como la tabla indexada. Si la posición [i,j] vale cierto, entonces en la fila *i*-ésima el atributo toma el valor *j*-ésimo.

Figura 48

Bolígrafos	Lápices	Portaláminas	Gomas	Hojas A4	Hojas A3	Tizas	Borradores		Cataluña	Castilla y León	Madrid	Andalucía
1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0

Ejemplo de mapas de bits

En la figura superior, podéis ver dos ejemplos de mapas de bits para la tabla de Hecho Ventas, uno para el atributo `Producto.nombre` y el otro para `Region.nombre`. Podemos ver que la primera y sexta filas de la tabla de ventas corresponden a ventas de bolígrafos (observando la primera columna del mapa de bits izquierdo), y que la cuarta corresponde a una venta en Madrid (observando la tercera columna del mapa de bits derecho).

Al contrario que los B^+ -árbol, los mapas de bits resultan especialmente interesantes para atributos con pocos valores posibles y una baja selectividad (el mismo valor repetido muchas veces). Se trata del índice ideal para utilizar en la tabla de clientes con un atributo como por ejemplo `sexo` (hombre o mujer), pero no tiene ningún sentido utilizarlo para `DNI`, porque todo el mundo tendrá un DNI distinto (obtendríamos una matriz muy grande en la que cada columna solo tendría un valor cierto).

Este tipo de índice ocupa muy poco espacio (si la tabla tuviera un millón de filas y el atributo ocho valores posibles, el índice ocuparía solo 1 Mbyte) y permite utilizar operadores lógicos de bajo nivel para resolver predicados sobre los atributos (solo hay que hacer AND y OR sobre secuencias de bits, que son operaciones muy rápidas). Además, se obtiene un rendimiento óptimo para resolver consultas que no necesitan acceder a los datos. Solo accediendo al índice, podemos contar cuántas filas cumplen o dejan de cumplir una cierta condición (contando ceros o unos, de manera respectiva). En los mapas de bits de la figura anterior podemos ver que tenemos una sola venta de borradores y cinco ventas en Cataluña, sin conocer el contenido de la tabla de hecho.

Los mapas de bits son fáciles de construir, mantener y utilizar.

Una posible manera de implementarlo consiste en definir un B^+ -árbol sobre una cierta Dimensión y guardar matrices de bits en las hojas del árbol que indiquen qué filas de la tabla de hecho apuntan al valor de esta tabla de Dimensión correspondiente a la hoja en cuestión. Os tenéis que fijar en la diferencia que hay con el apartado anterior, en el que definíamos un solo árbol para toda la tabla de hecho. Ahora tenemos un árbol diferente para cada tabla de Dimensión, cada uno de los cuales contiene mapas de bits de la tabla de Hecho. Con esto, si queremos hacer una consulta, utilizaremos el índice de cada Dimensión para obtener el mapa de bits correspondiente, operaremos con todos y accederemos directamente a las filas indicadas.

Figura 49

Bolígrafos	Lápices		Cataluña	Mapas de bits
1	0	1	1	1
0	0	0	1	0
0	1	1	0	0
0	0	0	0	0
0	OR	=	AND	=
1	0	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	1	0
0	1	1	1	1

Ejemplo de uso de los mapas de bits

En la figura 49, podéis ver cómo utilizaríamos los mapas de bits de las Dimensiones Producto y Lugar para saber qué ventas de bolígrafos y lápices se han hecho en Cataluña. Para obtener los datos, solo hay que acceder a las filas primera, sexta y décima.

Los índices de mapa de bits son especialmente apropiados para hacer consultas con muchas condiciones sobre diferentes atributos que tienen una selectividad baja. Si el atributo sobre el que definimos el índice tiene muchos valores posibles, la matriz resulta muy dispersa y derrocha mucho espacio.

6.3. Particiones horizontales

Como ya hemos dicho bastantes veces, una tabla de Hecho realmente es muy grande. Al mismo tiempo, acceder a la misma resulta imprescindible para resolver cualquier consulta. Por consiguiente, tenemos que facilitar el acceso a esta tabla tanto como podamos. Una manera de hacerlo consiste en dividirla en n subtablas o particiones, poniendo en cada una la enésima parte de las filas (es mejor que las particiones no se superpongan, que no tengan filas en común). Siempre podemos obtener fácilmente la tabla entera haciendo la unión de las particiones.

Cada partición puede estar en un disco diferente o, incluso, en una máquina distinta, con la correspondiente ganancia de rendimiento (aplicando técnicas de paralelismo). Como efecto colateral, también facilitamos la escalabilidad del sistema. Cuando necesitamos más espacio, no hace falta un disco donde podamos meter todos los datos, sino simplemente uno donde podamos meter como mínimo una de las particiones. El inconveniente en este caso es la disponibilidad. Bastará con que una sola máquina o disco no funcione para que no podamos responder la consulta.

Podemos definir las particiones según los valores de cualquier Dimensión (o incluso más de una Dimensión). Solo os tenéis que asegurar tan solo de que esta Dimensión no cambiará nunca. La reestructuración de todas las particiones puede generar problemas graves. Además, tened en cuenta que hacer particiones es especialmente útil (aunque no dispongamos de paralelismo, ni siquiera de discos diferentes), si una consulta solo tiene que acceder a una de las particiones o como mínimo no debe acceder a todas. Por lo tanto, tenemos que hacer particiones según un atributo que se acostumbre a fijar siempre en las consultas. Otra vez la fecha puede ser una buena elección. Podemos tener una partición para cada mes, cada semana o cada día si es preciso.

La partición horizontal de una tabla de Hecho facilita el paralelismo y la escalabilidad del sistema, a la vez que reduce el tiempo de respuesta para las consultas que utilicen para seleccionar los Descriptores de la Dimensión utilizada como criterio de partición.

6.4. Particiones verticales, herramientas VOLAP

Del mismo modo que podemos hacer una partición horizontal de la tabla del Hecho, para reducir su tamaño, también podemos hacerla verticalmente. En este caso, definimos distintas tablas con subconjuntos de los atributos, dando por sentado que cada uno de estos ha de incluir la clave primaria. Dado que habitualmente cada consulta contiene solo una Medida, lo podemos llevar al extremo y definir una tabla distinta para cada una, como podéis ver en la figura siguiente.

Figura 50

Ventas1(IDTiempo, IdCliente, IdProducto, IdLugar, IdPromocion)

Ventas2(IDTiempo, IdCliente, IdProducto, IdLugar, artículos)

La ganancia de la partición vertical es que maximizamos la proporción de datos útiles respecto de los datos a los que se ha accedido. Imaginemos que cada registro antes de la partición ocupa 28 bytes y, después de la partición, solo ocupa 24. Entonces, cada vez que accedemos a una página de disco, obtenemos aproximadamente un 16% más de valores de la Medida que nos interesa. Observemos, sin embargo, que prácticamente necesitamos el doble de espacio para tener Ventas1 y Ventas2 que el que se necesitaba antes para tener solo Ventas, sin particiones de ningún tipo.

Para mejorar el rendimiento del sistema, podemos definir particiones verticales de la tabla de hecho. Cada una de las particiones contendrá la clave primaria y una o más Medidas.

La mejora al aplicar esta técnica sería mucho más importante si no se tuviese que replicar la clave primaria en cada una de las tablas. En este caso, un registro ocuparía lo mínimo indispensable para guardar un valor, y todo el contenido de la página de disco a la que se ha accedido nos sería útil. Pues bien, hay herramientas OLAP que hacen exactamente esto. Estas herramientas reciben el nombre de VOLAP¹⁰.

⁽¹⁰⁾ Vertical OLAP.

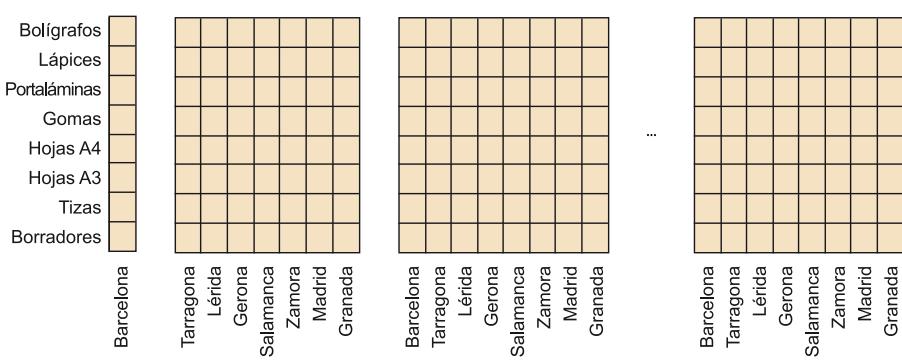
Las herramientas VOLAP ya no son relacionales. Se trata de SGBD específicos para análisis multidimensional que guardan las mediciones de una misma Medida todas seguidas sin los valores de la clave primaria, y que utilizan diferentes tipos de índice para localizarlos. Estos sistemas pueden obtener un tiempo de respuesta más de diez veces mejor que un SGBD relacional, y utilizan hasta cien veces menos espacio (aplican mecanismos de compresión). El problema que tienen es que se trata de soluciones totalmente propietarias, no estandarizadas, y que se basan en el supuesto de que solo queremos ver las Medidas de una en una.

Si las consultas solo piden valores de una sola Medida, podemos utilizar herramientas VOLAP.

6.5. Matrices n -dimensionales, herramientas MOLAP y HOLAP

Hay gente que cree que las bases de datos relacionales no son apropiadas para el análisis multidimensional y que utilizarlas es artificioso. De hecho, fueron concebidas para un mundo transaccional. Si lo que queremos consultar son cubos, ¿por qué almacenamos tablas? La respuesta son las herramientas *MOLAP* (*multidimensional OLAP*) o multidimensionales puras.

Figura 51

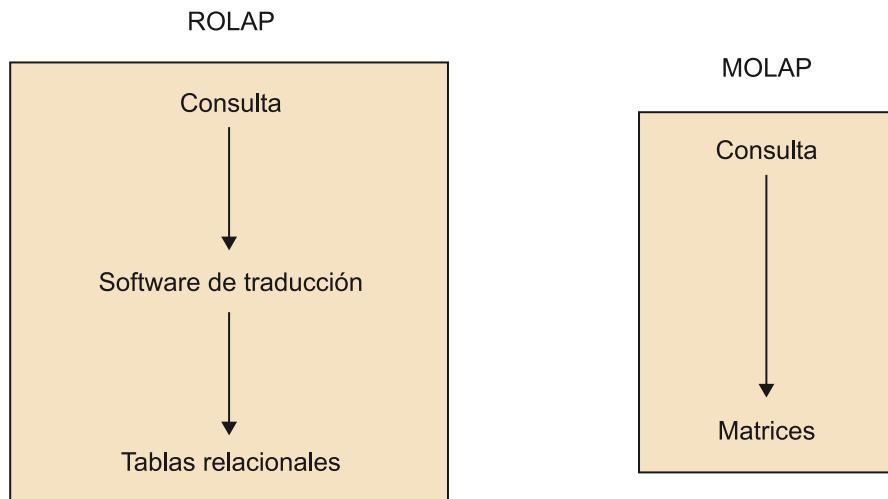


Las herramientas MOLAP almacenan matrices, como la que tenéis en la figura 51, e implementan sistemas de indexación especiales para acceder a las mismas. Ahora ya no hablamos de claves primarias, ni de claves foráneas. Cada elemento de la matriz contiene solo las Medidas. Estas matrices tienen exactamente la misma estructura que los cubos que queremos acabar visualizando. Es más, las matrices se definen en función de los cubos que serán consultados con más frecuencia. Es decir, dado el conjunto de consultas críticas (las

ejecutadas más a menudo y que piden un tiempo de respuesta más bajo), la herramienta MOLAP busca la manera de almacenar los datos para minimizar el tiempo de respuesta para estas consultas.

Las herramientas MOLAP utilizan como sistema de almacenamiento matrices n -dimensionales, en lugar de tablas relacionales.

Figura 52



En la figura 52 podéis ver la diferencia de filosofías entre una implementación ROLAP y una MOLAP. Mientras que la ROLAP necesita una traducción de la consulta a SQL, en una implementación MOLAP la consulta se resuelve directamente sobre los datos (no hay ningún paso intermedio).

Dado que no almacenan claves primarias, puede parecer que las herramientas MOLAP hacen lo mismo que las VOLAP. Esto no es cierto. La diferencia entre una herramienta MOLAP y una VOLAP es que la MOLAP almacena los datos según las consultas que se esperan, mientras que la VOLAP, no. En este sentido, una herramienta VOLAP está más cerca de una ROLAP que de una MOLAP.

Favorecer unas consultas más que otras es algo más natural de lo que parece. Por mucho que queramos almacenar matrices n -dimensionales, el disco solo tiene dos dimensiones (cilindros y sectores) y la memoria RAM, simplemente una. Por lo tanto, aunque una herramienta MOLAP intentara gestionar todas las Dimensiones de manera similar, no lo podría hacer. Habría alguna a la que se accedería de manera más eficiente que a las otras, porque los datos estarían físicamente agrupados. Por ejemplo, podríamos tener la información de los doce meses del año dentro del mismo sector y cilindro, de modo que accederíamos a todos sin necesidad de esperar toda la rotación de disco, ni mover el brazo para cambiar de cilindro. Por el contrario, si tenemos un mes en cada sector, deberemos esperar que el disco gire completamente para tenerlos todos.

Este sistema de almacenamiento con matrices dependientes de las consultas es muy eficiente, pero demasiado rígido. Pensad qué ocurre en la figura 51 si queremos añadir una nueva fecha. No hay ningún problema: simplemente añadimos a la derecha una nueva matriz de 8×8 . Sin embargo, ¿qué sucede si queremos añadir una nueva ciudad? Hay que reorganizar toda la matriz para tener sitio para poner los nuevos elementos allí donde les toca. Por ejemplo, tendríamos que abrir un agujero entre los datos de Granada y Barcelona para meter los nuevos elementos en medio. Además, añadir Dimensiones a una herramienta MOLAP multiplica el espacio de disco utilizado y resulta mucho más complejo que en una herramienta ROLAP, en la que simplemente hay que añadir un nuevo atributo a la clave de la tabla del Hecho.

Además del problema de la rigidez, encontramos el problema de las consultas no consideradas críticas o simplemente imprevistas. Puesto que el sistema no las ha tenido en cuenta para almacenar los datos, es bastante probable que tarde mucho en resolverlas.

Las herramientas MOLAP funcionan especialmente bien cuando tenemos pocas Dimensiones y muy estables (que nunca cambian). Dan muy buen resultado respecto al tiempo de respuesta, pero generalmente un mal resultado en lo que respecta al almacenamiento, especialmente para cubos con una dispersión grande. Esto se debe al hecho de que se guarda el cubo entero, tanto las celdas llenas como las vacías, para facilitar la indexación y el acceso. Para resolver el exceso de espacio utilizado, suelen utilizar técnicas de compresión de datos.

La principal ventaja de una herramienta MOLAP es su rapidez para consultar los datos. Sus puntos débiles son la gestión de grandes volúmenes de datos y la rigidez ante los cambios. Funcionan bien para almacenes de datos departamentales relativamente pequeños y estables.

Herramientas MOLAP

Algunas herramientas MOLAP intentan aprovechar más el espacio aplicando técnicas de compresión.

A pesar de que el tiempo de respuesta sea mejor con las herramientas MOLAP, las herramientas ROLAP se imponen en el mercado más por la misma implantación y el nivel de desarrollo de los sistemas relacionales que porque sean realmente adecuados para tareas multidimensionales. La estandarización del lenguaje SQL es un punto a favor muy importante de este tipo de herramientas, puesto que facilita la definición de traductores de consultas multidimensionales independientes de la SGBD. Sin embargo, además de la estandarización del lenguaje, también es cierto que las herramientas ROLAP demuestran una gran robustez y flexibilidad para tratar grandes volúmenes de datos como los de las tablas de hecho. Observad también que con la implementación sobre un SGBD relacional no tenemos problemas con la dispersión de los cubos. Solo hay filas para las celdas que contienen datos. Además, las herramientas MOLAP bajan mucho rendimiento cuando se hacen consultas imprevistas.

Para aprovechar lo mejor de las herramientas MOLAP y las ROLAP, también hay herramientas *HOLAP* (*hybrid OLAP*). Las herramientas MOLAP son mejores para cubos densos. Las herramientas ROLAP son mejores para cubos dispersos. Las herramientas HOLAP identifican regiones densas y dispersas y las almacenan según una técnica u otra. Cuando llega una consulta, la deshacen según la región del cubo involucrada y van a buscar los datos a un tipo de servidor u otro.

Las herramientas HOLAP mezclan lo mejor de las herramientas ROLAP y MOLAP.

6.6. Técnicas de preagregación

Como hemos visto hasta ahora, los diferentes tipos de herramientas multidimensionales aprovechan las distintas características de las consultas para mejorar el tiempo de respuesta. Las herramientas ROLAP aprovechan especialmente la división entre Hechos y Dimensiones; las VOLAP aprovechan que lo más habitual es pedir una sola Medida; las MOLAP, la repetición y previsión de consultas, etc. Pues bien, hay una característica del análisis multidimensional que aprovechan todas estas: la frecuencia de uso de las operaciones *roll-up* y *drill-down*. Sabemos que el usuario pedirá los datos a diferentes granularidades. Una manera de mejorar el tiempo de respuesta consiste en tener calculado el resultado de estas consultas antes de que lo pida.

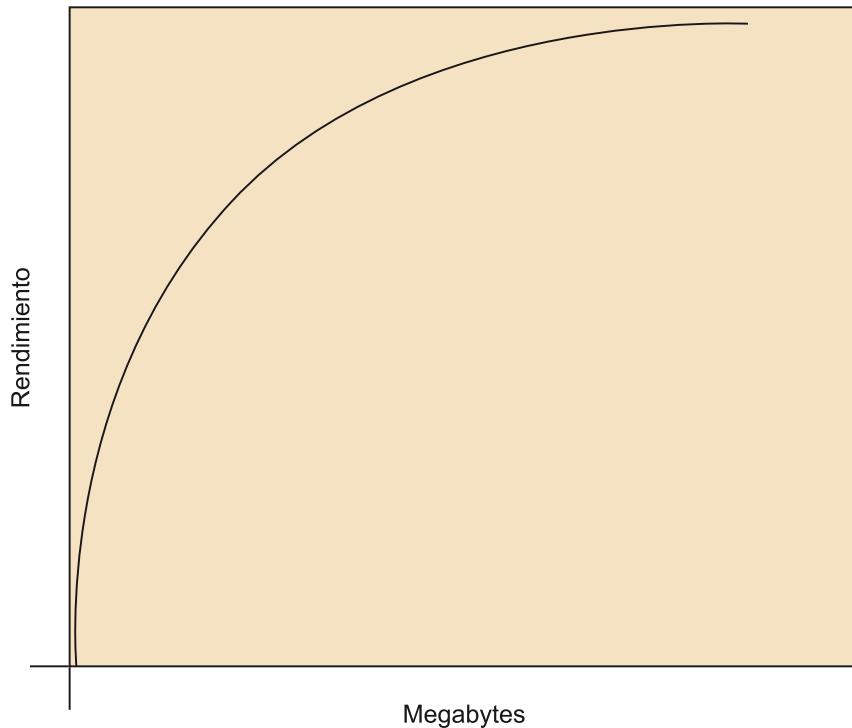
De manera independiente del tipo de herramienta que utilicemos, siempre se pueden aplicar técnicas de preagregación para mejorar el tiempo de respuesta ante las operaciones *roll-up* y *drill-down*. La preagregación es la herramienta más potente para mejorar el tiempo de respuesta de una aplicación multidimensional.

Recordad que en el apartado "Componentes del modelo multidimensional" ya hemos visto que cada Hecho contiene un retículo de Celdas relacionadas por agregaciones. A pesar de que en el diseño conceptual solo representamos algunas (las más importantes) de estas Celdas, el usuario las querrá ver todas en un momento u otro. Por lo tanto, el sistema tiene que garantizar que estos datos están disponibles y se pueden obtener con rapidez.

El coste de obtener datos agregados no viene dado por el cálculo que se tenga que hacer, sino por la simple obtención de los datos que hay que agregar.

La primera solución es guardar las instancias de todas las Celdas. Se trata de la solución más rápida (en cuanto al tiempo de respuesta) y la que derrocha más espacio. Observad que el número de Celdas crece de manera exponencial respecto al número de Dimensiones y Niveles. Simplemente, un Hecho con n Dimensiones con un Nivel para cada una tendrá 2^n Celdas. Desgraciadamente, esto hace que esta opción sea en la mayoría de los casos inviable.

En el extremo opuesto, podemos almacenar las instancias de la Celda de menor granularidad y calcular las instancias de las otras Celdas bajo demanda. Esta es la solución que ocupa menos espacio, pero también la más lenta. Lo que hay que hacer realmente es decidir qué Celdas guardamos físicamente (dicho de otro modo, materializadas) y cuáles calculamos solo cuando la consulta lo pida.



Como podéis ver en la gráfica, dedicando muy pocos Megabytes, mejoramos mucho el rendimiento. Sin embargo, llega un momento en el que, por más espacio que dediquemos a almacenar Celdas, no mejoramos nada el rendimiento del sistema.

Para tomar esta decisión, os tenéis que dar cuenta de que, cuanto más alta esté una Celda en el retículo, menos dispersión habrá. Por ejemplo, basta con que un día hayamos vendido algo para que, cuando agreguemos los meses, ya tengamos una celda para el mes correspondiente. Si nos imaginamos que solo tenemos datos de este mes, a granularidad Día, tenemos una dispersión 31/1 (solo uno de los treinta y un puntos del espacio contiene una celda), mientras que, a granularidad Mes, tenemos una dispersión 1/1 (el único punto del espacio contiene una celda). En un caso como este, necesitaríamos el mismo número de tuplas para almacenar los datos detallados y para almacenar los

agregados, y no conseguiríamos ninguna ganancia en el tiempo de respuesta. Cuanto más dispersos sean los datos básicos, más espacio ocuparán de manera proporcional los datos agregados.

Antes de almacenar físicamente una Celda, nos tenemos que asegurar de que cada una de sus instancias resulta de la agregación de, como mínimo, diez instancias de la Celda a partir de la que la calculamos. Si no, la pequeña ganancia de tiempo de respuesta no compensará el gasto de espacio extra.

El espacio que necesitaríamos para almacenar todas las Celdas es de órdenes de mayor magnitud que el que ocupan los datos básicos. La experiencia dice que tenemos que dedicar aproximadamente el mismo espacio para datos preagregados que el que ocupan los datos básicos.

Ya hemos tomado una primera decisión al hacer el diseño conceptual. Las Celdas que hemos explicitado (porque contienen Medidas que no podemos calcular a partir de las Medidas de las otras Celdas o porque el usuario las considera especialmente importantes) son las primeras candidatas para almacenar. Ahora solo hay que ver cuál de las otras queremos almacenar junto a estas. Por ejemplo, si almacenamos ventas mensuales (que denotaremos `Ventas(Cliente, Producto, Poblacion, Mes)`), ya no es preciso que consultemos las ventas diarias (`Ventas(Cliente, Producto, Poblacion, Dia)`) para calcular las ventas anuales (`Ventas(Cliente, Producto, Poblacion, Año)`), porque es mucho más económico hacerlo a partir de las mensuales.

Decidir qué datos hace falta preagregar para obtener un tiempo de respuesta cercano al que obtendríamos preagregándolos todos, pero dedicando un espacio de disco y un tiempo de actualización razonables, dependerá de muchos factores (por ejemplo, el hardware disponible, las características de la red y del software, el número de usuarios, etc.). Además, nunca encontraremos el conjunto perfecto de Celdas que hay que preagregar, porque las consultas de los usuarios evolucionan con el paso del tiempo.

El problema de elegir qué Celdas materializamos no resulta nada trivial (teniendo en cuenta su complejidad computacional, se trata de un problema NP-completo). Sin embargo, una buena solución es ordenarlas según su utilidad. Las Celdas más útiles son las que sirven para resolver las consultas más costosas y comunes. Una Celda es útil tanto si la pide el usuario con mucha frecuencia, como si sirve para obtener fácilmente la que pide a menudo el usuario. Una Celda no es muy útil si ya hemos decidido almacenar las instancias de otra Celda a partir de la cual ya podemos obtenerla fácilmente. Una vez hecha esta ordenación, si sabemos el espacio que ocupa cada Celda y el que tenemos disponible, aplicamos un algoritmo voraz (*greedy*) y almacenamos tantas Celdas

como podamos, siguiendo el orden de utilidad fijado anteriormente. Algunos sistemas ya hacen la elección de manera automática, una vez el administrador indica el espacio de disco que le quiere dedicar.

Al decidir el espacio que queremos dedicar a guardar datos preagregados, hemos de tener en cuenta que será necesario actualizar los datos agregados cada vez que modifiquemos la Celda atómica. Observad que esto puede hacer crecer mucho la ventana de actualización. Por lo tanto, para decidir el espacio dedicado, no solo tenemos que mirar el precio del disco, sino el tiempo necesario para mantener actualizados estos datos preagregados.

La mejor estrategia para decidir qué datos tenemos preagregados es el ensayo error.

7. Ejemplo de diseño multidimensional

Para consolidar los conceptos expuestos en los apartados 3, 4 y 6, vamos a desarrollar paso a paso un modelo multidimensional a partir de un caso.

Supongamos que nuestra organización brinda atención telefónica a su cartera de clientes y quiere contabilizar el tiempo invertido en atención de llamadas por mes y por cliente. Concretamente vamos a considerar el análisis de las llamadas recibidas por el centro de atención al cliente, considerando una serie de variables como tiempo, localización y motivo.

Específicamente y como mínimo:

- número de llamadas en un periodo concreto,
- evolución de la duración de las llamadas,
- motivos más repetidos en una zona geográfica determinada.

La compañía telefónica nos remite la información periódicamente en documentos con el formato de la figura 53:

Figura 53. Ejemplo de datos

Número de llamada	Fecha	Ciudad	Zona	Duración (minutos)	Motivo
327441	10/12/2015	París	Europa	8	Queja
327442	21/01/2016	Ámsterdam	Europa	5,5	Avería
327443	24/01/2016	Nueva York	América	14	Queja
327444	07/02/2016	Tokio	Asia	4,75	Sugerencia
...					
n	28/03/2016	Rabat	África	9	Queja

7.1. Diseño conceptual

El modelo conceptual se basa en identificar qué tipo de procesos y vistas de negocio proporcionan respuesta a las preguntas de los usuarios finales. Normalmente, en esta fase, se debe ser previsor y pensar más allá de las necesidades actuales y poder cubrir las futuras.

En el apartado 3 de este módulo, se detallan los pasos que hay que seguir: elegir el hecho, encontrar el gránulo oportuno, elegir las dimensiones que se utilizarán en el análisis, encontrar los atributos de cada dimensión, etc.

Procedemos a identificar los dos tipos de elementos principales:

- **Dimensiones:** representan factores a través de los cuales se analiza una determinada área del negocio.
- **Hechos:** son el objeto de análisis y están relacionados con las dimensiones.

Teniendo en cuenta la información disponible de las llamadas telefónicas, identificamos para nuestro ejemplo una tabla de hecho o proceso de negocio: **la llamada**.

Cada llamada puede analizarse desde diferentes puntos de vista (lo que nos proporciona las dimensiones del proceso de negocio):

- Dimensión territorio: ciudad desde la que se realiza la llamada.
- Dimensión motivo: información de la causa que origina la llamada.
- Dimensión fecha: momento en el que se realiza la llamada.

Los hechos contienen los datos de estudio (número de llamada y duración) y las dimensiones contienen los metadatos sobre dichos hechos (cuándo, dónde y por qué).

Si la información necesita disponer de varios niveles de granularidad, se crean jerarquías con las dimensiones. Así tenemos las jerarquías siguientes:

- Para fecha podría ser «día – mes – trimestre – año».
- Para territorio podría ser «ciudad – país – zona».

Pero ¿qué hay de su representación? Igual que sucede en el modelo relacional, el modelo dimensional adopta el concepto de relación entre tablas como estructura básica del modelo. Pero a diferencia del modelo relacional, que no hace distinción entre relaciones, el dimensional distingue entre relaciones de hecho (tablas de hecho) y relaciones de dimensión (tablas de dimensión).

Veremos que existe una tabla de hechos en el centro y alrededor tablas de dimensiones, una para cada dimensión de análisis que participa de la descripción del hecho.

Dada la distribución de los elementos en el gráfico, nuestro modelo multidimensional se asemeja a una estrella y se denomina **modelo en estrella**. Si tuviera dimensiones relacionadas con otras dimensiones, se denominaría **modelo copo de nieve**.

De esta manera, presentamos en la figura 54 el diseño conceptual de nuestro ejemplo, que adopta el modelo en estrella:

Figura 54. Diseño conceptual



7.2. Diseño lógico

El punto de partida para diseñar el modelo lógico será la estrella obtenida en el diseño conceptual. El objetivo de esta fase es identificar las métricas del hecho y los atributos de las dimensiones. Si hubiéramos obtenido más de una estrella, llevaríramos a cabo el diseño lógico estrella a estrella.

Para implementar una estrella, necesitamos una tabla para el hecho (en la que cada fila representa una celda del espacio multidimensional) y una tabla más para cada una de las dimensiones.

La tabla de hecho contiene la clave subrogada que identifica de manera única cada registro, las claves foráneas a las dimensiones relacionadas con la tabla de hecho y las métricas. En nuestro caso existe una única métrica que es la duración de llamada.

Las jerarquías de agregación quedan implícitas en los valores de los atributos de las tablas de dimensión. No los explicitamos con tablas diferentes.

Cuando a una dimensión no se le pueden asociar múltiples atributos, se dice que tenemos una dimensión degenerada, y solo aparecerá como una columna en la tabla de hechos. Suele tener su origen en atributos identificadores de las bases de datos operacionales que coinciden con el gránulo elegido por la estrella en cuestión. Siempre que sea posible es importante conservarlos.

En nuestro caso, la dimensión degenerada es el número que identifica la llamada. A pesar de no tener ningún atributo, nos resulta útil para identificarla. Esta dimensión llamada no tiene ningún atributo, porque para definir las otras dimensiones de análisis se los hemos sacado todos (fecha, territorio, etc.). Y no va a existir en nuestro diseño porque vamos a trasladar el número que identifica la llamada a las métricas de la tabla de hecho.

De esta manera, observamos en la figura 55, las características que tenemos para la tabla de hecho **h_llamada**:

Figura 55. Tabla de hecho

Tabla de hecho	Claves foráneas	Métricas
h_llamada	id_temporal, id_teritorio, id_motivo	Número de llamada Duración de la llamada

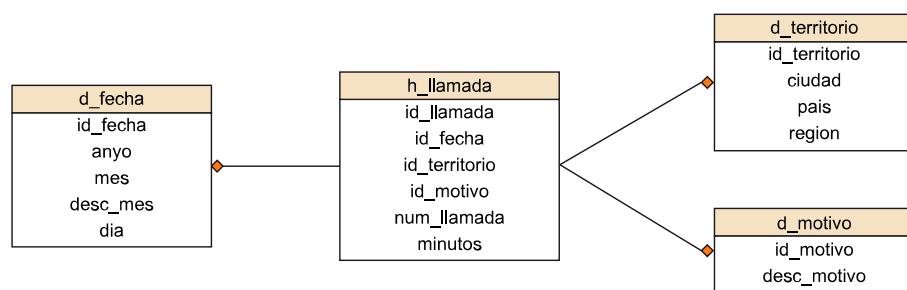
En la tabla 56, observamos los atributos de cada una de las dimensiones:

Figura 56. Tabla de dimensiones

Dimensión	Clave primaria	Atributos
d_teritorio	id_teritorio	ciudad, país y región
d_fecha	id_fecha	año, mes, desc_mes y día
d_motivo	id_motivo	desc_motivo

En la figura 56 se presenta el diseño lógico resultante:

Figura 56. Diseño lógico



7.3. Diseño físico

El último paso que nos falta para completar el proceso de creación del modelo de datos multidimensional es el diseño físico. Para llevarlo a cabo, trabajaremos con una base de datos Oracle (eXpress Edition) que será usada para el *Data Warehouse* y con la herramienta de modelización de base de datos: Oracle SQL Developer.

Así, nuestra opción es un modelo ROLAP, que nos permite un buen desempeño del sistema en base al uso de índices para las claves de todas las tablas y particiones horizontales (por años, por ejemplo).

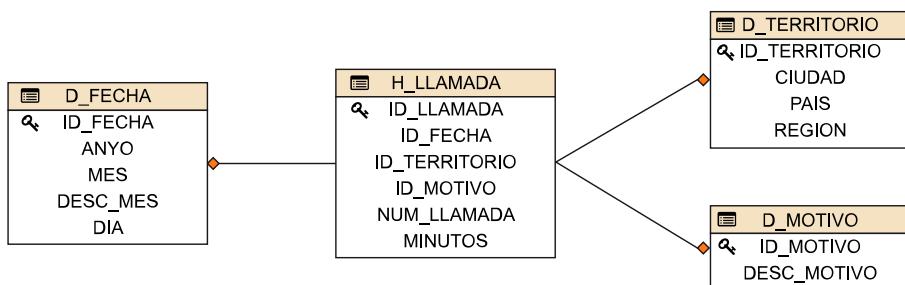
Debemos recordar que las herramientas MOLAP funcionan especialmente bien cuando tenemos pocas dimensiones y muy estables (que nunca cambian). Permiten obtener muy buen tiempo de respuesta, pero generalmente un mal resultado en lo que respecta al volumen de almacenamiento.

El objetivo es definir, para cada tabla, el formato de cada clave y atributo. Debemos recordar los siguientes criterios:

- Se recomienda que las claves sean enteras y que sean independientes de las fuentes de origen.
- Las métricas pueden ser aditivas (números), semiaditivas (con particularidades en el momento de acumular las cantidades) o no aditivas (atributos cualitativos). En las tablas de hecho deberíamos decantarnos porque todas las métricas fueran de las aditivas.
- Es necesario incluir campos que permitan la trazabilidad del dato, por ejemplo, fecha de carga, fecha de modificación, autor, fuente de origen. Para simplificar el modelo, no se incluyen.

Si consideramos cada una de las tablas y el detalle de cada uno de los atributos que la componen, entonces obtenemos un diseño físico como el que se muestra en la figura 57:

Figura 57. Diseño físico



Es posible presentar diseños alternativos al modelo propuesto.

Se muestran bajo estas líneas, los *scripts* de creación de las tablas necesarias para materializar el diseño físico finalmente propuesto:

```

CREATE TABLE d_territorio
(
  id_territorio INTEGER
, ciudad VARCHAR2(25)
, pais VARCHAR2(15)
, Zona VARCHAR2(15)
, constraint "D_TERRITORIO_PK" primary key (id_territorio)
);
  
```

```

CREATE TABLE d_fecha
(
  id_fecha INTEGER
, fecha DATE
, anyo VARCHAR2(4)
, mes VARCHAR2(2)
, dia VARCHAR2(10)
, constraint "D_FECHA_PK" primary key (id_fecha)
);
  
```

```
) ;
```

```
CREATE TABLE d_motivo
(
    id_motivo INTEGER
, desc_motivo VARCHAR2(25)
, constraint "D_MOTIVO_PK" primary key (id_motivo)
);
```

```
CREATE TABLE d_territorio
(
    id_territorio INTEGER
, ciudad VARCHAR2(25)
, pais VARCHAR2(15)
, Zona VARCHAR2(15)
, constraint "D_TERRITORIO_PK" primary key (id_territorio)
);
```

7.4. Definición del cubo OLAP en Mondrian

Veremos a continuación la implementación del modelo que acabamos de diseñar en un servidor OLAP, concretamente en Mondrian, que es el motor OLAP integrado en la suite de Pentaho.

Mondrian se caracteriza por ser un motor ROLAP con memoria caché, lo cual lo sitúa cerca del concepto de HOLAP. ROLAP significa que en Mondrian no residen datos (salvo en la caché) sino que estos están en una base de datos en la que existen las tablas (definidas en el diseño físico) que conforman la información multidimensional con la que el motor trabaja.

Mondrian se encarga de recibir las consultas dimensionales a un cubo mediante MDX y de devolver los datos. El cubo es, en este caso, un conjunto de metadatos que definen como se ha de mapear la consulta por sentencias SQL al repositorio que contiene realmente los datos.

En la siguiente imagen podemos ver la sintaxis de nuestro cubo:

Figura 58

```

<Schema name="LLAMADAS">

    <Dimension type="StandardDimension" visible="true" highCardinality="false" name="Motivo">
        <Hierarchy name="Motivo" visible="true" hasAll="true" primaryKey="id_motivo">
            <Table name="d_motivo"/>
            <Level name="Motivo" visible="true" column="Motivo" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
        </Hierarchy>
    </Dimension>

    <Dimension type="StandardDimension" visible="true" highCardinality="false" name="Territorio" primaryKey="id_fecha">
        <Hierarchy name="Territorio" visible="true" hasAll="true" primaryKey="id_fecha">
            <Table name="d_fecha"/>
            <Level name="Ciudad" visible="true" column="CIUDAD" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
            <Level name="Pais" visible="true" column="PAIS" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
            <Level name="Region" visible="true" column="REFION" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never" />
        </Hierarchy>
    </Dimension>

    <Dimension type="TimeDimension" visible="true" highCardinality="false" name="Fecha">
        <Hierarchy name="Fecha" visible="true" hasAll="true" primaryKey="id_territorio">
            <Table name="d_territorio"/>
            <Level name="Anyo" visible="true" column="Anyo" type="String" uniqueMembers="false" levelType="TimeYears" hideMemberIf="Never" />
            <Level name="Mes" visible="true" column="mes" type="String" uniqueMembers="true" levelType="TimeMonths" hideMemberIf="Never" />
            <Level name="Dia" visible="true" column="dia" type="String" uniqueMembers="true" levelType="TimeDays" hideMemberIf="Never" />
        </Hierarchy>
    </Dimension>

    <Cube name="VENDES" visible="true" cache="true" enabled="true primaryKey="id_llamada">
        <Table name="H_LLAMADA"/>
        <DimensionUsage source="Motivo" name="Motivo" visible="true" highCardinality="false" foreignKey="id_motivo"/>
        <DimensionUsage source="Territorio" name="Territorio" visible="true" highCardinality="false" foreignKey="id_territorio"/>
        <DimensionUsage source="Fecha" name="Fecha" visible="true" highCardinality="false" foreignKey="id_fecha"/>
        <Measure name="Quantitat" column="ID_LLAMADA" aggregator="count" visible="true"/>
        <Measure name="Minutos" column="MINUTOS_LLAMADA" aggregator="sum" visible="true" />
        <CalculatedMember name="Promig" formula="[Measures].[Minutes]/[Measures].[Quantitat]" dimension="Measures" visible="true" />
    </Cube>
</Schema>
```

Web de interés

Se puede consultar la información completa de los esquemas Mondrian en: Mondrian Documentation.

8. Consultas con MDX

El lenguaje de consulta OLAP es MDX (*MultiDimensional eXpressions* o expresiones multidimensionales), creado por Microsoft en 1997 para poder formular problemas OLAP de forma fácil.

Este lenguaje ha sido acogido por la gran mayoría de los proveedores de herramientas OLAP y se ha convertido en norma de hecho para estos sistemas. Hoy en día es el lenguaje más usado para consultas sobre cubos.

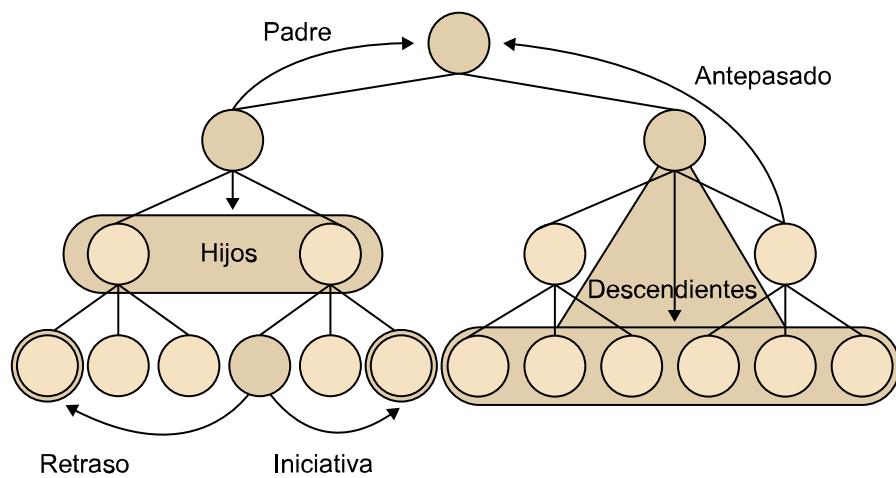
El lenguaje MDX es, en los sistemas OLAP, el equivalente al SQL. Aunque es posible traducir algunas de sus sentencias a SQL tradicional, con frecuencia se requieren expresiones SQL poco claras incluso para las sentencias más simples del MDX.

Algunas de sus principales características son:

- Permite explotar la información que reside en los motores OLAP y satisfacer las consultas analíticas.
- Tiene funciones y fórmulas que lo hacen muy potente para el análisis de datos.
- Las jerarquías permiten a este lenguaje poder referenciar los diferentes elementos de las dimensiones con expresiones del tipo: MIEMBROS-HIJO, MIEMBROS-PRIMO, MIEMBROS-PADRE, etc.

En la figura 59 vemos una representación gráfica de las posibles relaciones:

Figura 59. Posibles relaciones entre miembros



8.1. Estructura básica de la consulta

La sintaxis de MDX es compleja. La estructura general de una consulta MDX tiene la forma:

```
SELECT
{ elementos } ON COLUMNS,
{ elementos } ON ROWS
FROM [ nombre_cubo ]
WHERE { elementos }
```

- En el *select* se especifica el conjunto de elementos que queremos visualizar que debe detallarse si se devuelve en columnas y/o filas.
- En el *from*, el cubo de donde se extrae la información.
- En el *where*, las condiciones de filtrado.
- {} permite crear listas de elementos en las selecciones.
- [] encapsula elementos de las dimensiones y niveles.

Ejemplo

Imaginemos un cubo de ventas con las siguientes dimensiones:

- Temporal de las ventas con niveles de año y mes.
- Territorios en los que vendemos con niveles de región y país.
- Medidas: importe de las ventas y unidades vendidas.

Para obtener las unidades vendidas en Francia para el año 2005, la consulta sería:

```
SELECT
{ [Medidas].[unidades] } ON COLUMNS,
{ [Region].[Pais].[Francia] } ON ROWS
FROM [cubo ventas]
WHERE{ [Tiempo].[2005] }
```

8.2. Tipos de elementos

A continuación vamos a recordar algunos elementos básicos con los que tendremos que estar familiarizados para generar nuestras consultas MDX:

- **Dimensión:** una dimensión es una colección de atributos que se relacionan mediante una jerarquía y que se relacionan a su vez con los hechos de la dimensión de medidas.
- **Atributo o nivel de dimensión:** un atributo de dimensión está enlazado a una o más columnas de una tabla de dimensiones y contiene miembros. Un atributo de dimensión puede contener nombres de clientes, de meses o de productos.

- **Miembro:** un miembro es un valor de un atributo de dimensión, incluida la dimensión de medidas. Un miembro de una jerarquía puede ser un nivel hoja, un miembro primario, un miembro de datos o un miembro (All).
- **Medida:** una medida es un valor de una tabla de hechos; también se denomina hecho. Por lo general, también se hace referencia a un valor de la dimensión de medidas como miembro. Las medidas suelen ser valores numéricos, pero también pueden ser valores de cadena.
- **Dimensión de medidas:** una dimensión de medidas es la dimensión que contiene todas las medidas de un cubo. Una dimensión de medidas es un tipo especial de dimensión en la que los miembros se suelen agregar (generalmente mediante la suma o el recuento) de acuerdo con el miembro actual de cada atributo de dimensión con el cual existe una medida especificada.
- **Miembro calculado:** un miembro calculado es un miembro de dimensión que se define y calcula en tiempo de consulta. Un miembro calculado puede definirse en una consulta de usuario o en el *script* de cálculo MDX; se almacena en el servidor. Un miembro calculado corresponde a las filas de la tabla de dimensiones de la dimensión en la que se definen.
- **Tupla:** una tupla identifica de forma exclusiva a una celda según una combinación de miembros. Una tupla puede estar compuesta de uno o más miembros de la dimensión medidas, aunque solo uno de cada una de las demás dimensiones del cubo.
- **Conjunto o set:** un conjunto o *set* es un conjunto ordenado de tuplas con la misma dimensionalidad. Se utilizan llaves {} para designar un conjunto de tuplas.

Ejemplos de Miembros, Hijos(Children) y Descendants:

a) {[Time].[2005].[Month].Members} ¿Qué nos devolverá?

Nosotros veremos: Enero, Febrero, Marzo...

Realmente:

```
[2005].[Enero], [2005].[Febrero], [2005.Marzo]...
```

b) ¿Y para obtener los cuatrimestres (Q1, Q2 ...) del año 2005?

```
{ [Time].[2005].[Quarter].Members}
```

c) ¿Y qué obtendríamos con { [Time]. Children}?

```
2010, 2011, 2012, 2013...
```

d) Para obtener los descendientes de un nivel se usa: Descendants(Miembro, Nivel)

```
Descendants( [Time].[2005] , [Time].[Month] )
```

8.3. Funciones básicas

Tiene similitudes con el lenguaje SQL, aunque incluye funciones y fórmulas especiales orientadas al análisis de estructuras jerarquizadas que presentan relaciones entre los diferentes miembros de las dimensiones.

Los nombres de funciones son palabras reservadas del lenguaje MDX. Las más habituales son:

a) **NON EMPTY**: elimina del resultado de la consulta los valores nulos. Se antepone a las medidas o dimensiones.

Ejemplo

```
NON EMPTY( { [Mesaures].primera_medida} )
```

b) **CurrentMember**: permite acceder al miembro actual.

c) **Children**: permite acceder a todos los hijos de una jerarquía.

d) **prevMember**: permite acceder al miembro anterior de la dimensión.

e) **Hierarchize**: ordena los miembros de un conjunto en una jerarquía. Se antepone a las medidas o dimensiones.

Ejemplo

```
Hierarchize( [Mesaures].primera_medida)
```

f) **CrossJoin**: realiza *join* de dos o más conjuntos (normalmente dimensiones). También se expresa con un signo de multiplicación (*).

Ejemplo

```
CrossJoin( [Dim1]. Miembros, [Dim2]. Miembros )
[Dim1]. Miembros * [Dim2].Miembros
```

g) **BottomCount** (conjunto_datos,N): permite obtener, de un conjunto de datos ordenado de forma ascendente, el número especificado de tuplas(N) con los valores más bajos.

h) **BottomSum** (conjunto_datos,N,S): obtiene de un conjunto ordenado los N elementos cuyo total es como mínimo el especificado (S).

i) **Except** (conjunto_datos1, conjunto_datos2): permite obtener la diferencia entre dos conjuntos de datos.

j) **Union** (conjunto_datos1, conjunto_datos2,...): une varios conjuntos de datos, eliminando los duplicados.

k) AVG, COUNT, VARIANCE y todas las funciones trigonométricas (seno, coseno, tangente, etc.).

l) PeriodsToDate (nivel, miembro): devuelve a un conjunto los miembros del mismo nivel que el miembro indicado, empezando por el primer miembro del mismo nivel y acabando con el miembro en cuestión, de acuerdo con la restricción del nivel especificado en la dimensión de tiempo.

También son de uso frecuente algunas funciones especiales de «tiempo»:

- WTD(<Miembro>): devuelve los miembros de la misma semana del miembro especificado.
- MTD(<Miembro>): devuelve los miembros del mismo mes del miembro especificado.
- QTY(<Miembro>): devuelve los miembros del mismo trimestre del miembro especificado.
- YTD(<Miembro>): devuelve los miembros del mismo año del miembro especificado.
- ParallelPeriod(<Nivel>, <Expresión numérica>, <Miembro>): devuelve un miembro de un periodo anterior en la misma posición relativa que el miembro especificado.

Por otro lado, interesa insistir en la importancia de los miembros calculados en MDX.

Una de las funcionalidades más potentes que ofrece el lenguaje MDX es la posibilidad de realizar cálculos complejos dinámicos (en función de los datos que se están analizando en ese momento) y estáticos. Los cubos multidimensionales trabajan con medidas (*measures*) y con miembros calculados (*calculated members*).

Las medidas son las métricas de la tabla de hechos a las que se aplica una función de agregación (*count*, *distinct count*, *sum*, *max*, *avg*, etc.).

Un miembro calculado es una métrica que tiene como valor el resultado de la aplicación de una fórmula que puede utilizar todos los elementos disponibles en un cubo, así como otras funciones de MDX disponibles. Estas fórmulas admiten desde operaciones matemáticas hasta condiciones semáforicas pasando por operadores de condiciones.

Se definen previamente al SELECT de la consulta MDX:

Ejemplo

```
WITH member [Measures].[Mi_calculo] AS
'[Measures].[primera_medida] / [Measures].[segunda_medida]'
SELECT [Measures].[Mi_calculo] ON COLUMNS,
[Territorio].members ON ROWS
FROM [cubo]
```

8.4. Consultas simples

A continuación veremos algunos ejemplos que ilustran el uso de las funciones anteriormente descritas, sobre los datos del ejemplo del apartado anterior.

Para llevarlos a cabo hemos usado el visor Saiku (*versión community*) que ejecuta consultas sobre una base de datos multidimensional (Mondrian).

Saiku surge como remplazo del visor JPivot pero, en lugar de fundamentarse y replicar JPivot, es un desarrollo completamente nuevo. Sus principales características son las siguientes:

- Es un visor con una huella en memoria pequeña.
- Está orientado al usuario por lo que permite aplicar todo tipo de funciones del lenguaje MDX de forma visual y mediante *drag & drop*.
- Soporta múltiples bases de datos y no solo Mondrian como motor OLAP.

Ejemplo. ¿Cuántas llamadas de cada tipo se han recibido?

Para responder a la pregunta del título, una posible consulta sería:

```
SELECT
[Measures].[Cantidad] ON COLUMNS,
[Motivo].[Causa].Members ON ROWS
FROM [LLAMADAS]
```

En la figura 60 vemos el resultado de ejecutar la consulta:

Figura 60. Resultado de la ejecución de la consulta MDX

The screenshot shows the Saiku Analytics interface. At the top, there's a toolbar with File, View, Tools, Help, and a dropdown menu labeled 'Opened'. Below the toolbar is a navigation bar with tabs for Cubes, Measures, and Dimensions. The 'Cubes' tab is selected, showing a list with 'LLAMADAS' highlighted. The 'Measures' tab shows 'Cantidad' as the selected measure. The 'Dimensions' tab lists 'Fecha', 'Motivo', and 'Territorio'. On the right side, there's a code editor window displaying the MDX query:

```
1 SELECT
2 [Measures].[Cantidad] ON COLUMNS,
3 [Motivo].[Causa].Members ON ROWS
4 FROM [LLAMADAS]
5
```

Below the code editor, the status '2, 20' is shown. To the right of the code editor is a data grid titled 'Causa' with the following data:

Causa	Cantidad
Avería	4
Queja	5
Sugerencia	3

Ejemplo. Usando la cláusula NON EMPTY: ¿Cuántas llamadas de cada tipo se han recibido?

Para responder a la pregunta del título, una posible consulta sería:

```
SELECT
NON EMPTY [Measures].[Minutos] ON COLUMNS,
NON EMPTY [Motivo].[Causa].Members ON ROWS
FROM [LLAMADAS]
```

Al ejecutarla obtenemos el resultado que se muestra en la figura 61:

Figura 61. Resultado de la consulta MDX

Causa	Minutos
Avería	29.5
Queja	64
Sugerencia	18.4

Ejemplo. ¿Cuántas llamadas se han recibido cada año en cada ciudad?

Para responder a la pregunta del título, una posible consulta sería:

```
SELECT
[Fecha].[Anyo].Members ON COLUMNS,
[Territorio].[Ciudad].Members ON ROWS
FROM [LLAMADAS]
```

Al ejecutar la consulta, se puede observar en el resultado que se muestra en la figura 62 que, al no haber indicado ninguna medida en la consulta, el motor nos devuelve directamente el número de hechos que corresponden a esa celda.

Figura 62. Resultado de la consulta MDX

```

1 SELECT
2   [Fecha].[Año].Members ON COLUMNS,
3   [Territorio].[Ciudad].Members ON ROWS
4  FROM [LLAMADAS]
    
```

Ciudad	2015	2016
Amsterdam		1
Buenos Aires		1
Lisboa		1
Londres		1
Madrid		1
Manila		1
Nueva Delhi		1
Nueva York		1
Paris	1	
Rabat		1
Tokio		1
Toronto		1

Ejemplo. ¿Cuántos minutos hemos dedicado a atender las llamadas de los distintos países según la causa que las originó?

Para responder a la pregunta del título, una posible consulta sería:

```

SELECT
NON EMPTY { [Measures].[Minutos] } ON COLUMNS,
NON EMPTY { [Motivo].[Causa].Members } * {[Territorio].[Pais].Members}
ON ROWS
FROM [LLAMADAS]
    
```

Al ejecutarla obtenemos el resultado de la figura 63:

Figura 63. Resultado de la consulta MDX

```

1 SELECT
2   NON EMPTY {[Measures].[Minutos]} ON COLUMNS,
3   NON EMPTY {[Motivo].[Causa].Members} * {[Territorio].[Pais].Members} ON ROWS
4  FROM [LLAMADAS]
    
```

Causa	País	Minutos
Avería	Holanda	5.5
	Portugal	6.5
	Reino Unido	9
	España	8.5
Queja	Argentina	17
	Estados Unidos	14
	Francia	8
	Marruecos	10
	Canadá	15
Sugerencia	Filipinas	5.8
	India	7.8
	Japón	4.8

Ejemplo. ¿Cuál es el total de minutos por mes y año que hemos destinado a cada tipo de llamadas?

Para responder a la pregunta del título, una posible consulta sería:

```
SELECT
NON EMPTY CrossJoin (Hierarchize({{[Fecha].[Anyo].Members},
{[Fecha].[Mes].Members}}),
{[Measures].[Minutos]}) ON COLUMNS,
NON EMPTY [Motivo].[Causa].Members ON ROWS
FROM [LLAMADAS]
```

Al ejecutarla obtenemos el resultado de la figura 64:

Figura 64. Resultado de la consulta MDX

The screenshot shows the Saiku Analytics interface. At the top, there's a menu bar with File, View, Tools, and Help. Below the menu is a toolbar with various icons. The main workspace is titled "Saiku Analytics". On the left, there's a sidebar with sections for Cubes (LLAMADAS selected), Measures (Add), Dimensions (Fecha, Motivo), and Fact (Cantidad, Minutos, Promedio). The central area displays the MDX query and its results. The MDX code is:

```
1 SELECT
2 NON EMPTY CrossJoin(Hierarchize({{{[Fecha].[Anyo].Members},
3 {[Fecha].[Mes].Members}}}),
4 {[Measures].[Minutos]}) ON COLUMNS,
5 NON EMPTY [Motivo].[Causa].Members ON ROWS
6 FROM [LLAMADAS]
```

Below the code is a table with the following data:

Causa	2015		2016					
	12		1	2	3	4	5	
Avería			29.5	5.5	6.5		17.5	
Queja	8	8	56	14		25	17	
Sugerencia			18.4		4.8	5.8	7.8	

9. Nuevas tendencias

En este apartado recogemos las últimas tendencias según los diferentes tipos de motores OLAP. Principalmente difieren en cómo guardan los datos:

a) **Extreme OLAP**: este nombre se está empezando a usar en la industria para referirse a un motor OLAP que trabaja sobre alguna de las tecnologías *Big Data* (como Hadoop o Spark).

b) **Desktop OLAP**: es un caso particular de OLAP ya que está orientado a equipos de escritorio. Consiste en obtener la información necesaria desde la base de datos relacional y guardarla en el escritorio. Las consultas y análisis son realizados contra los datos guardados en el escritorio.

c) **In-memory OLAP**: es un enfoque por el que muchos fabricantes están optando. Consisten en que la estructura dimensional se genera solo en la memoria y se guarda el dato original en algún formato que potencia su despliegue de esta forma (por ejemplo, comprimido o mediante una base de datos de lógica asociativa). En este último punto es donde cada fabricante pone su énfasis.

No podemos finalizar sin presentar HTAP (*Hybrid Transaction/Analytic Processing*). En 2014 Gartner acuñó este acrónimo para describir una nueva tecnología que soporta el uso operacional (OLTP) y el uso analítico (OLAP) sin necesidad de ninguna infraestructura adicional.

HTAP permite la detección en tiempo real de tendencias a las que podemos dar una rápida respuesta. Por ejemplo, puede permitir a los minoristas identificar rápidamente los libros más vendidos en la última hora e inmediatamente crear ofertas personalizadas para ellos.

Las bases de datos convencionales no son capaces de soportar HTAP. Sin embargo, la aparición de tecnologías NewSQL similares en rendimiento y escalabilidad a las tecnologías NoSQL, pero con las propiedades ACID propias de las bases de datos tradicionales, permite habilitar esta capacidad híbrida para manejar OLTP, OLAP y otras consultas analíticas.

Empresas de bases de datos como MemSQL, VoltDB, NuoDB y InfinitumDB ofrecen soluciones comerciales que implementan HTAP.

Hadoop

Apache Hadoop es un framework para el procesamiento y almacenamiento de grandes volúmenes de datos.

Spark

Apache Spark es un motor para el procesamiento rápido de datos a gran escala.

Resumen

A veces, para desarrollar un cierto tipo de aplicaciones, tenemos que aplicar técnicas específicas de diseño. Este es el caso de las herramientas OLAP, utilizadas para hacer el análisis multidimensional. Su diseño se basa en la definición de Hechos objeto de análisis y las Dimensiones utilizadas para analizarlos.

En este módulo, hemos visto cuáles son los elementos principales de un modelo multidimensional: qué estructuras de datos ofrece, qué operaciones podemos hacer con los datos y qué restricciones de integridad podemos definir. También hemos estudiado cómo se hace un diseño conceptual multidimensional paso a paso y cómo se puede implementar en un SGBD relacional. Para acabar, habéis conocido algunas técnicas de acceso para mejorar el tiempo de respuesta y las nuevas palabras reservadas definidas en la SQL'99 para facilitar la escritura de las consultas.

Actividades

1. Leed las dieciocho reglas de evaluación de herramientas multidimensionales del Dr. E. F. Codd. Las podéis encontrar en el libro de E. Thomsen *OLAP Solutions*, o también en <http://www.olapreport.com>.
2. Podéis ver cómo un cierto SGBD (por ejemplo, SQL Server de Microsoft) implementa las operaciones ROLLUP y CUBE del estándar SQL'99.

Ejercicios de autoevaluación

1. Haced el diseño multidimensional para el caso de un conjunto de almacenes de productos de un distribuidor de supermercados. Este distribuidor sirve como intermediario entre los proveedores y las tiendas. Estamos especialmente interesados en analizar lo siguiente:

- a) La cantidad de cada producto (*stock*) que tenemos cuando acaba el día en cada uno de los almacenes, teniendo en cuenta que podemos haber comprado el mismo producto a distintos proveedores.
- b) Cada movimiento que hay en los almacenes (independientemente de si es una entrada – qué nos sirve un proveedor– o una salida –hacia una tienda–), según el producto, la hora en que tiene lugar y la entidad (ya sea proveedor o tienda) que lo genera.

Para cada movimiento tenemos un albarán con un código que lo identifica de manera única, asignado por el sistema operacional de gestión de los almacenes común a toda la empresa. Un albarán contiene los datos del almacén correspondiente, la hora de entrega y el nombre del proveedor o tienda, junto con la lista de productos que se sirven.

Cada almacén (del mismo modo que los proveedores y las tiendas) está en una población y cada población pertenece a una región. Tanto almacenes como tiendas y proveedores se identifican por un nombre. Para los almacenes, también disponemos de los metros cuadrados de superficie.

De la Dimensión temporal, nos interesan los meses, los trimestres, los cuatrimestres y los años. Además de una hora, día, mes, trimestre, cuatrimestre o año cualquiera, también queremos poder seleccionar los días festivos y los meses de verano.

Los productos están identificados por el código de barras, pero también queremos ver su descripción. Interesa agrupar los productos en tipos de producto, que se identifican por su nombre. Pensemos que un producto no puede ser de más de un tipo.

2. Estimad el espacio que ocupa el esquema anterior. Pensad que estamos interesados en los datos correspondientes a los últimos tres años (mil días aproximadamente), nuestro catálogo contiene 10.000 productos diferentes (divididos en 1.000 tipos), tenemos diez almacenes, servimos a 1.000 tiendas y compramos a 1.000 fabricantes. De media, a cada fabricante le hacemos un pedido semanal, mientras que cada tienda nos hace un pedido día sí, día no. En total, al acabar el año tenemos 235.000 albaranes, con una media de 50 productos por albarán.

3. Dado el esquema y los volúmenes de datos anteriores, decid qué tipo de herramienta OLAP sería mejor utilizar (ROLAP, MOLAP, etc.).

4. Traducid el esquema anterior a relacional.

5. Pensad ahora que para mejorar el tiempo de respuesta de las consultas en la relación *StockDiario*(Producto,Día,Region,cantidad,stock) que contiene el *stock* de los 10.000 productos (de 1.000 tipos), los 1.000 días, a los 10 almacenes (repartidos en 3 regiones), también decidimos almacenar las tres relaciones siguientes, que contienen datos preagregados de las Celdas correspondientes:

Stock1(Producto,Día,Region,cantidad,stock)
Stock2(Producto,Año,Almacen,cantidad,stock)
Stock3(Tipo,Año,Almacén,cantidad,stock)

a) ¿A cuántas tuplas hay que acceder para resolver una consulta que pide el *stock* anual de un cierto producto en una región si lo calculamos accediendo a *Stock*? ¿A cuántas accediendo a *Stock1*? ¿A cuántas accediendo a *Stock2*? ¿A cuántas accediendo a *Stock3*?

b) ¿Cuál de las cuatro relaciones deberíamos utilizar para saber el total de artículos (de todos los productos) que hemos tenido en *stock* en todos los años en todas las regiones?

c) ¿Qué Celdas podemos calcular a partir de Stock3?

6. Encontrad una secuencia de operaciones multidimensionales que os permitan pasar del primer cubo al segundo. Los dos pertenecen a la Estrella `Inventario` diseñada anteriormente.

Origen				
Cantidades de cada tipo de producto por albarán	1.111	2.222	3.333	4.444
Materiales de oficina	1	2	3	4
Alimentos	4	3	2	1

Destino				
Cantidades de producto por región de almacén y día	23-4-2002	24-4-2002	25-4-2002	26-4-2002
Cataluña	Gomas	1	0	0
	Bolígrafos	0	2	3
				2

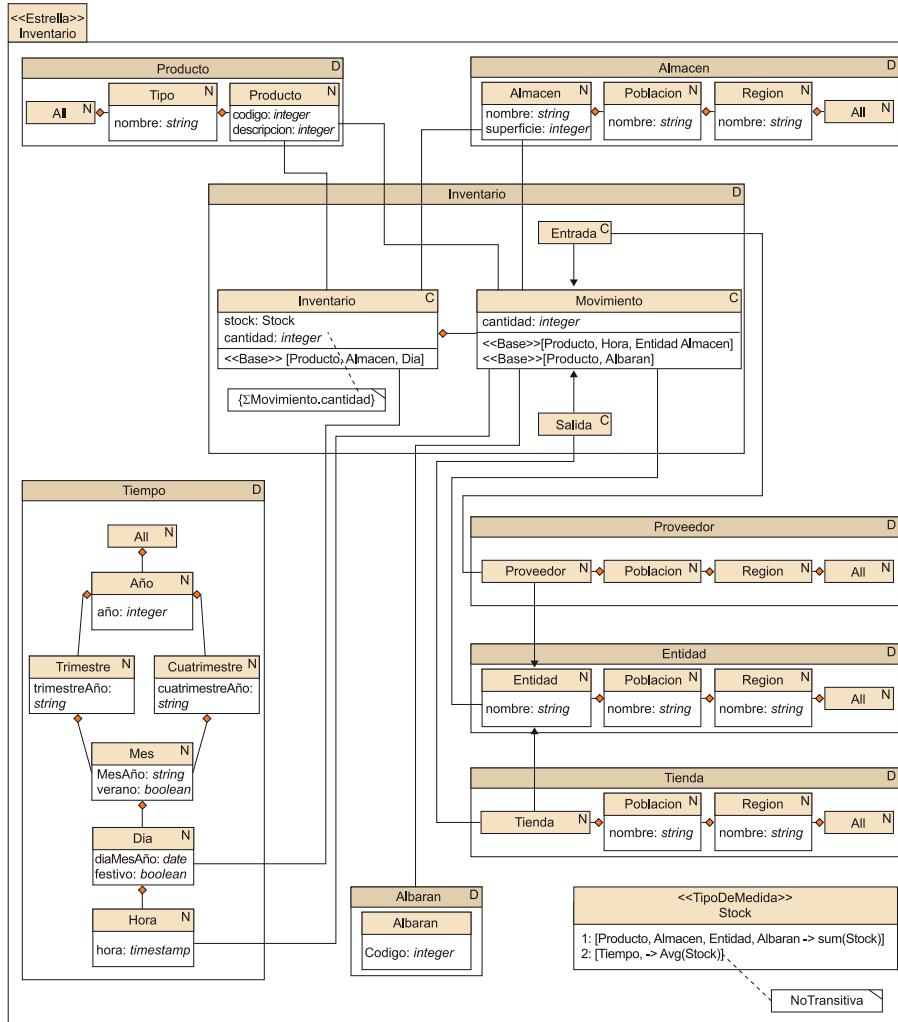
7. Haced una consulta de la tabla Entrada definida en el ejercicio 4, utilizando las palabras reservadas definidas en el estándar de 1999, para obtener el resultado siguiente escribiendo lo mínimo posible:

Región de almacén	Año	Tipo de producto	Cantidad
Cataluña	2001	Materiales de oficina	100
Cataluña	2001	Alimentos	200
Cataluña	2001	Null	300
Cataluña	2002	Materiales de oficina	400
Cataluña	2002	Alimentos	500
Cataluña	2002	Null	900
Cataluña	Null	Materiales de oficina	500
Cataluña	Null	Alimentos	700
Cataluña	Null	Null	1.200

Solucionario

Ejercicios de autoevaluación

1. La cantidad que hay en el almacén puede provenir de distintos fabricantes. Por lo tanto, no podemos asociar la Dimensión Fabricante con la Celda StockMensual.



2. Un hito máximo del espacio de la Celda Movimiento es 4.800.000.000.000 celdas (10.000 productos \times 24.000 horas \times 2.000 entidades \times 10). Ahora bien, podemos obtener un hito más pequeño si lo calculamos a partir del número de albaranes. De este modo, obtenemos que el espacio tiene como máximo 7.050.000.000 celdas (3 años \times 235.000 albaranes/año \times 10.000 productos). Sin embargo, dado que el enunciado dice que hay una media de 50 productos en cada albarán, sabemos que realmente serán solo 35.250.000 celdas. Si pensamos que cada cantidad ocupa 2 bytes y que necesitaremos cinco identificadores (uno para cada Dimensión) de 4 bytes cada uno, la Celda ocuparía 775.500.000 bytes (775,5 Mb).

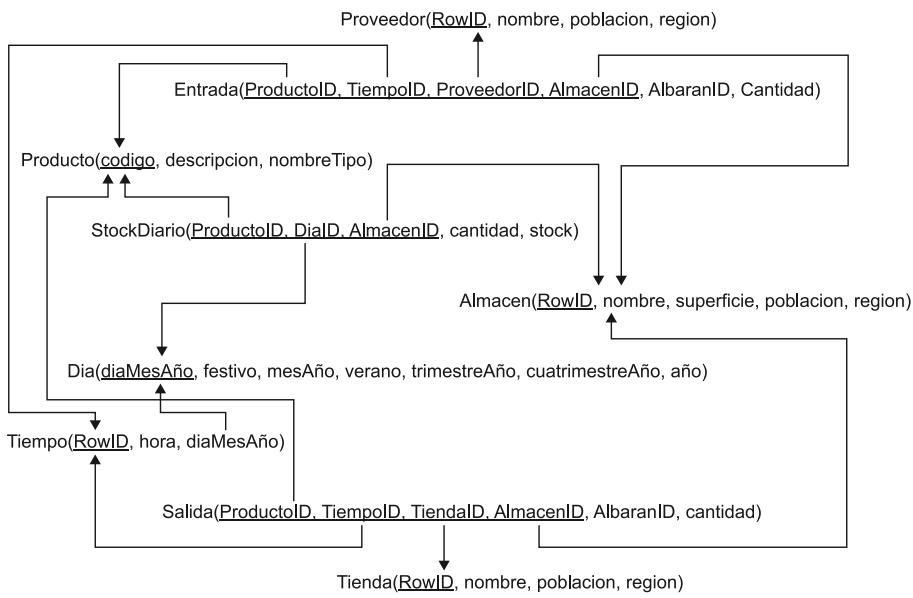
La Celda StockDiario define un espacio máximo de 100.000.000 celdas (10.000 productos \times 10 almacenes \times 1.000 días). Pensemos que cada stock ocupa 2 bytes y que necesitaremos tres identificadores de 4 bytes cada uno. La Celda ocuparía 1.600.000.000 bytes (1.600 Mb).

En total, toda la Estrella ocupará 2375,5 Mb (si negligimos el espacio que puedan ocupar las Dimensiones).

3. Los datos correspondientes a los movimientos son muy dispersos. A cada hora no nos llegan pedidos de todos los clientes ni nos sirven todos los proveedores. Probablemente, la mejor elección sería una herramienta ROLAP, por el gran volumen de datos que soportan y porque no tienen problemas con la dispersión. Por el contrario, la dispersión del cubo correspondiente a StockDiario es prácticamente 1. Todas las celdas que puede haber casi siempre estarán presentes. Una herramienta MOLAP que soporte este volumen de datos sería

la mejor opción, porque probablemente nos ofrecerá un tiempo de respuesta menor que una implementación ROLAP. Si queremos tener los dos cubos dentro del mismo sistema, deberíamos elegir un HOLAP.

4. Albaran es una Dimensión degenerada, que quedará simplemente como parte de una clave alternativa. No desaparece del todo, porque nos puede interesar hacer consultas por albarán. Además de esto, dado que las subclases de Movimiento no tienen ningún atributo específico, en lo que respecta al espacio que ocupará, es equivalente tener una superclase o dos subclases. Por lo tanto, nos tenemos que fijar simplemente en el tiempo de respuesta. ¿Será más rápido consultar una tabla grande o dos pequeñas? En este caso, implementaremos la especialización como si fuera una partición horizontal. Cada tipo de movimiento está en una tabla distinta (si los queremos todos, solo hay que hacer la unión).



5.

a) En el peor caso, será necesario acceder a todas las tuplas de cada relación. **StockDiario** contiene 10^8 tuplas, **Stock1** contiene 3×10^7 y **Stock2** contiene 3×10^5 . **Stock3** contiene solo 3×10 tuplas, pero no lo podemos utilizar porque sus datos no son lo bastante detallados (están en granularidad tipo de productos y queremos consultar productos).

b) A pesar de que sería mucho más eficiente utilizar **Stock2** o **Stock3**, tendríamos que usar **Stock1** si queremos tener el resultado exacto. Recordad que a lo largo del tiempo agregamos utilizando la media y que la media no es una función transitiva. De este modo, tenemos que acceder a los datos en el nivel más detallado, que en este caso es **Día**.

c) Considerando que no podemos continuar agregando a lo largo de la Dimensión **Tiempo** por la razón mencionada anteriormente, y que los tipos de producto son disyuntos (como dice el enunciado), podríamos calcular las Celdas siguientes:

Stock4(Tipo, Año, Poblacion, cantidad)
Stock5(Tipo, Año, Region, cantidad)
Stock6(Tipo, Año, cantidad)
Stock7(Año, Almacen, cantidad)
Stock8(Año, Poblacion, cantidad)
Stock9(Año, Region, cantidad)
Stock10(Año, cantidad)

6. Una posible solución podría ser la siguiente:

$A := \text{Selección}_{\text{Tipo.nombre} = "Materiales de oficina"}(\text{origen})$
 $B := \text{CambioBase}_{\text{Producto} \times \text{Almacén} \times \text{Tiempo} \times \text{Entidad}}(A)$
 $C := \text{Drill-down}_{\text{Producto}::\text{Producto}}(B)$
 $D := \text{Roll-up}_{\text{Entidad}::\text{Alj}}(C)$
 $E := \text{Roll-up}_{\text{Almacén}::\text{Región}}(D)$
 $F := \text{Roll-up}_{\text{Tiempo}::\text{Día}}(E)$
 $\text{destino} := \text{CambioBase}_{\text{Almacén} \times \text{Producto} \times \text{Tiempo}}(F)$

7.

```
SELECT m.region, d.año, p.tipo, SUM(f.cantidad)
FROM entrada f, almacen m, tiempo t, dia d, producto p
WHERE f.AlmacenID=m.RowId
AND f.TiempoID=t.RowID
AND t.diaMesAño=d.diaMesAño
AND f.ProductoID=p.codigo
AND d.año IN (2001,2002)
AND p.tipo IN ("Alimentos", "Materiales de oficina")
AND m.region="Cataluña"
GROUP BY m.region, CUBE(d.año, p.tipo)
ORDER BY m.region, d.año, p.tipo;
```

Glosario

almacén de datos corporativo *m* Conjunto de datos integrados e históricos de toda la empresa.

almacén de datos departamental *m* Conjunto de datos que resuelve las necesidades de análisis de un cierto departamento o conjunto de usuarios.

dato *m* Medición. Observación hecha y almacenada en algún sistema.

descriptor *m* Atributo de una dimensión utilizado para seleccionar instancias de los hechos.

dice *f* Operación multidimensional que reduce el tamaño del espacio que selecciona valores en las dimensiones.

dimensión *f* Punto de vista utilizado en el análisis de un cierto hecho.

dispersión *f* Cociente entre el número máximo teórico de instancias que puede llegar a haber y las instancias que hay realmente en un cubo.

drill-across *m* Operación multidimensional que, dado un espacio, cambia los datos que se muestran. Cambia de un tema de análisis a otro.

drill-down *m* Operación multidimensional que muestra los datos más detallados.

entidad-interrelación

sigla ER

en entity-relationship

factoría de información corporativa *f* Conjunto de elementos de software y hardware que ayudan en el análisis de datos para tomar decisiones.
sigla FIC

granularidad *f* Tamaño de un objeto respecto a otro.

hecho *m* Objeto de análisis.

HOLAP *m* Herramienta OLAP que mezcla características ROLAP y MOLAP.
en hybrid olap

información *f* Datos relevantes para algún decisor que afectan a alguna de sus decisiones.

jerarquía de agregación *f* Conjunto de relaciones entre las instancias de una dimensión que indica cómo agrupamos algunas de estas para obtener otras.

medida *f* Dato numérico asociado a un acontecimiento que queremos analizar.

metadato *m* Dato sobre los datos.

MOLAP *m* Herramienta OLAP que utiliza matrices *n*-dimensionales (en lugar de tablas relacionales) para almacenar los datos.
en multidimensional OLAP

nivel de agregación *m* Conjunto de instancias de la misma granularidad dentro de una dimensión.

O³LAP *m* Herramienta OLAP implementada sobre un SGBD orientado al objeto.
en object-oriented OLAP

OLAP *m* Siglas que hacen referencia a las herramientas de análisis, normalmente multidimensional. Categoría de tecnología de software que permite a los analistas, gestores y ejecutivos mejorar su conocimiento de los datos mediante el acceso rápido, consistente e interactivo a una amplia variedad de posibles vistas de información que ha sido transformada desde los datos operacionales para reflejar la dimensionalidad real de la empresa como la entiende el usuario (The OLAP Council).
en on-line analytical processing

OLTP *m* Siglas que hacen referencia a sistemas operacionales, que ayudan en el día a día de nuestra empresa.

en on-line transactional processing

ROLAP *m* Herramienta OLAP implementada sobre un SGBD relacional.

roll-up *m* Operación multidimensional que resume los datos aumentando la granularidad a lo largo de una Dimensión.

SGBD *m* Véase sistema de gestión de bases de datos.

sistema de gestión de bases de datos *m* Software que gestiona y controla bases de datos. Sus funciones principales son las de facilitar el uso simultáneo a muchos usuarios de distintos tipos, independizar al usuario del mundo físico y mantener la integridad de los datos.

sigla **SGBD**

en database management system

sistema operacional *m* Sistema que ayuda en las operaciones diarias de negocio de una organización.

sistema transaccional *m* Sistema basado en transacciones de lectura/escritura.

slice *f* Operación multidimensional que reduce la dimensionalidad del espacio fijando un valor en una dimensión.

ventana de actualización *f* Periodo de tiempo dedicado a la actualización de un almacén de datos.

VOLAP *m* Herramienta OLAP que almacena los datos por columnas en lugar de por filas.

en vertical OLAP

Bibliografía

Kimball, R. (2010). *The Kimball Group Reader; Relentlessly Practical Tools for data warehousing and Business Intelligence*. Indianapolis: John Wiley & Sons, Inc.

Mendack, S. (2008). *OLAP without cubes: Data Analysis in non-cube Systems*. Hoboken: VDM Verlag.

Webb, C. y otros (2006). *MDX Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase*. Hoboken: John Wiley & Sons.

Wrembel, R. (2006). *Datawarehouses and OLAP: Concepts, Architectures and Solutions*. Hershey: IGI Globals.

<http://www.kimballgroup.com>.