

Diseño de análisis OLAP

Josep Curto Díaz

PID_00243064

Índice

Introducción	5
1. OLAP como herramienta de análisis	7
1.1. Tipos de OLAP	8
1.2. Elementos OLAP	10
1.3. Doce reglas OLAP de E. F. Codd	11
2. OLAP en el contexto de Pentaho	13
2.1. Mondrian	13
2.2. Visores OLAP	15
2.3. Herramientas de desarrollo	20
3. Caso práctico	24
3.1. Diseño OLAP con Pentaho Schema Workbench	24
3.2. Publicación de un esquema OLAP en Pentaho Server	34
Abreviaturas	39
Bibliografía	40
Anexo	41

Introducción

Es bien sabido que el concepto de *business intelligence* engloba múltiples conceptos. Uno de los más importantes es el concepto OLAP (*online analytical processing*), acuñado por Edgar F. Codd.

Una manera sencilla de entender qué significa este concepto es que se trata de una tecnología que permite el análisis multidimensional a través de tablas matriciales o pivotantes.

Si bien el término *OLAP* se introduce por primera vez en 1993, sus conceptos base, como por ejemplo el análisis multidimensional, son mucho más antiguos.

A pesar de ser una tecnología que ya tiene más de cuatro décadas, sus características y su evolución han producido que la gran mayoría de las soluciones del mercado incluyan un motor OLAP.

Es necesario comentar las siguientes cuestiones:

- Las herramientas OLAP de los diferentes fabricantes, si bien son similares, no son completamente iguales, dado que presentan diferentes especificaciones del modelo teórico.
- Las últimas tendencias en OLAP incluyen la tecnología *in-memory* así como su adaptación a tecnologías *big data*.
- Las soluciones *open source* OLAP han sido las últimas en añadirse a la lista y, por ahora, no tienen tanta variedad como su contrapartida propietaria.
- En el mercado *open source* OLAP solo hay dos soluciones actualmente: el motor ROLAP Mondrian y el motor MOLAP PALO.
- Están apareciendo soluciones OLAP para *big data*, como Apache (creado por eBay) o Pinot (creado por LinkedIn).

Este módulo se centrará en presentar el concepto OLAP y sus diferentes opciones.

Análisis multidimensional

En 1962 se introdujo el análisis multidimensional en la obra de Ken Iverson *A Programming Language*.

1. OLAP como herramienta de análisis

OLAP forma parte de lo que se conoce como sistemas analíticos que permiten responder a preguntas como ¿por qué pasó? Estos sistemas pueden encontrarse tanto integrados en *suites* de *business intelligence* o ser simplemente una aplicación independiente.

Es necesario, antes de continuar, introducir una definición formal de OLAP:

Se entiende por **OLAP**, o **proceso analítico en línea**, el método para organizar y consultar datos sobre una estructura multidimensional. A diferencia de las bases de datos relacionales, todas las potenciales consultas están calculadas de antemano, lo que proporciona una mayor agilidad y flexibilidad al usuario de negocio.

Una herramienta OLAP está formada por un motor y un visor. El motor es, en realidad, justo el concepto que acabamos de definir. El visor OLAP es una interfaz que permite consultar, manipular, reordenar y filtrar datos existentes en una estructura OLAP mediante una interfaz gráfica de usuario que dispone de funciones de consulta MDX y otras.

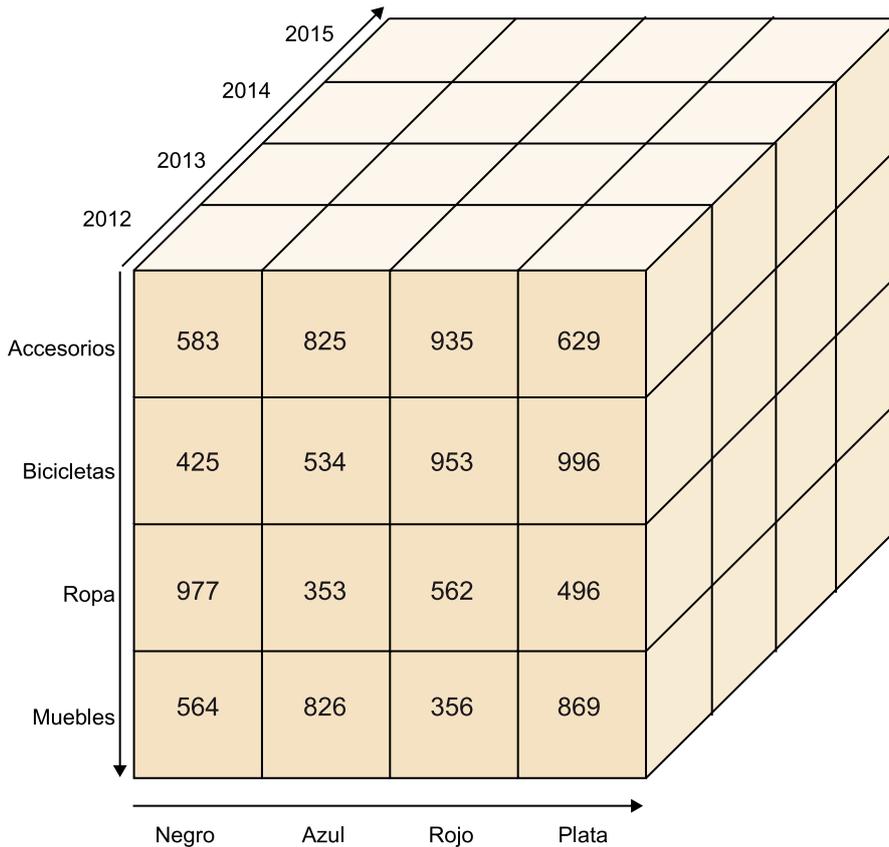
Las estructuras OLAP permiten realizar preguntas que serían sumamente complejas mediante SQL.

Consideremos un ejemplo gráfico que nos permitirá entender la potencia de este tipo de herramientas.

MDX

Es el lenguaje de consulta sobre estructuras multidimensionales.

Figura 1. Ejemplo de cubo OLAP



Fuente: Josep Curto.

Imaginemos que queremos responder a la siguiente pregunta: ¿cuál es el margen de beneficios de la venta de bicicletas en el año 2014?

Si tenemos un cubo, como el de ejemplo, formado por el año, los productos y su gama de colores, la respuesta es la intersección entre los diferentes elementos.

Cabe observar que una estructura de esta forma permite consultas mucho más completas, como por ejemplo comparar el margen de beneficios de 2013 y 2014, entre diferentes productos, etc.

Además, mediante el visor OLAP se proporciona libertad a los usuarios finales para realizar dichas consultas de manera independiente al departamento de IT.

1.1. Tipos de OLAP

Existen diferentes tipos de OLAP, que principalmente difieren en cómo se guardan los datos:

1) **MOLAP (multidimensional OLAP)**: es la forma clásica de OLAP y frecuentemente es referida con dicho acrónimo. MOLAP utiliza estructuras de bases de datos generalmente optimizadas para la recuperación de estos, lo que se conoce como bases de datos multidimensionales (o más coloquialmente cu-

bos). En definitiva, se crea un fichero que contiene todas las posibles consultas precalculadas. A diferencia de las bases de datos relacionales, estas formas de almacenaje están optimizadas para la velocidad de cálculo. También se optimizan a menudo para la recuperación a lo largo de patrones jerárquicos de acceso. Las dimensiones de cada cubo son típicamente atributos tales como periodo, localización, producto o código de la cuenta. La forma como cada dimensión será agregada se define por adelantado.

2) **ROLAP (*relational OLAP*)**: trabaja directamente con las bases de datos relacionales, que almacenan los datos base y las tablas dimensionales como tablas relacionales, mientras que tablas nuevas se crean para guardar la información agregada.

3) **HOLAP (*hybrid OLAP*)**: no hay acuerdo claro en la industria en cuanto a qué constituye el OLAP híbrido, exceptuando el hecho de que es una base de datos en la que los datos se dividen entre almacenaje relacional y multidimensional. Por ejemplo, para algunos vendedores, HOLAP consiste en utilizar las tablas relacionales para guardar las cantidades más grandes de datos detallados, y utiliza el almacenaje multidimensional para algunos aspectos de cantidades más pequeñas de datos menos detallados o agregados.

4) **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).

5) **DOLAP (*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.

6) **In-memory OLAP**: es un enfoque por el que muchos fabricantes están optando. La estructura dimensional se genera a nivel de memoria solo, y se guarda el dato original en algún formato que potencie 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.

Cada tipo tiene ciertas ventajas, aunque hay desacuerdo sobre las ventajas específicas entre los diferentes proveedores.

- MOLAP es mejor en sistemas más pequeños de datos, es más rápido para calcular agregaciones y retornar respuestas, y necesita menos espacio de almacenaje. Últimamente, *in-memory OLAP* está apuntalándose como una opción muy válida al MOLAP.
- ROLAP se considera más escalable. Sin embargo, es difícil implementar eficientemente el preproceso de grandes volúmenes de datos, lo que hace

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.

que se desechen con frecuencia. De otro modo, el funcionamiento de las consultas podría ser no óptimo.

- HOLAP está entre los dos en todas las áreas, pero puede preprocesar rápidamente y escalar bien.
- *Extreme* OLAP será usado cuando la empresa implemente proyectos de *big data* y estén interesados en tener disponible análisis OLAP sobre grandes volúmenes de datos.

Todos los tipos son, sin embargo, propensos a la explosión de la base de datos, exceptuando *extreme* OLAP. Este es un fenómeno que causa la cantidad extensa de espacio de almacenaje que es utilizado por las bases de datos OLAP cuando se resuelven ciertas, pero frecuentes, condiciones: alto número de dimensiones, de resultados calculados de antemano y de datos multidimensionales escasos.

La dificultad en la implementación OLAP radica en la formación de las consultas, elegir los datos base y desarrollar el esquema. Como resultado, la mayoría de los productos modernos proveen de bibliotecas enormes de consultas preconfiguradas. Otro problema que puede afectar al cubo es trabajar con datos de baja calidad o incompletos

1.2. Elementos OLAP

OLAP permite el análisis multidimensional, lo que significa que la información está estructurada en ejes (puntos de vista de análisis) y celdas (valores que se están analizando).

En el contexto OLAP existen diferentes elementos comunes a las diferentes tipologías OLAP (que en definitiva se diferencian a nivel práctico en que en MOLAP se precalculan los datos, en ROLAP no y en *in-memory* se generan al iniciar el sistema):

- **Esquema:** es una colección de cubos, dimensiones, tablas de hecho, métricas y roles.
- **Cubo:** es una colección de dimensiones asociadas a una tabla de hecho. Un cubo virtual permite cruzar la información entre tablas de hecho a partir de sus dimensiones comunes.
- **Tabla de hecho:** representa un proceso de negocio que analizar.
- **Dimensión:** representa las diferentes perspectivas de análisis de un proceso de negocio.

- **Métrica:** representa el resultado de un proceso de negocio.
- **Jerarquía:** es un conjunto de miembros organizados en niveles. Desde el punto de vista de las bases de datos se puede entender como una ordenación de los atributos de una dimensión.
- **Nivel:** es un grupo de miembros en una jerarquía que tienen los mismos atributos y nivel de profundidad en la jerarquía.
- **Miembro:** es un punto en la dimensión de un cubo que pertenece a un determinado nivel de una jerarquía. Las métricas (medidas) en OLAP se consideran un tipo especial de miembro que pertenece a su propio tipo de dimensión. Un miembro puede tener propiedades asociadas.
- **Roles:** permisos asociados a un grupo de usuarios.
- **MDX:** es un acrónimo de *multidimensional expressions* (aunque también es conocido como *multidimensional query expression*). Es el lenguaje de consulta de estructuras OLAP, fue creado en 1997 por Microsoft y si bien no es un lenguaje estándar la gran mayoría de los fabricantes de herramientas OLAP lo han adoptado como estándar de hecho.

1.3. Doce reglas OLAP de E. F. Codd

La definición de OLAP presentada anteriormente se basa en las doce leyes que acuñó Edgar F. Codd en 1993. Estas reglas son las que, en mayor o menor medida, intentan cumplir todos los fabricantes de software:

- 1) **Vista conceptual multidimensional:** se trabaja a partir de métricas de negocio y sus dimensiones.
- 2) **Transparencia:** el sistema OLAP debe formar parte de un sistema abierto que soporta fuentes de datos heterogéneas (lo que llamamos actualmente arquitectura orientada a servicios).
- 3) **Accesibilidad:** se debe presentar el servicio OLAP al usuario con un único esquema lógico de datos (lo que en definitiva nos indica que debe presentarse respecto a una capa de abstracción directa con el modelo de negocio).
- 4) **Rendimiento de informes consistente:** el rendimiento de los informes no debería degradarse cuando el número de dimensiones del modelo se incrementa.
- 5) **Arquitectura cliente/servidor:** basado en sistemas modulares y abiertos que permitan la interacción y la colaboración.

6) **Dimensionalidad genérica:** capacidad de crear todo tipo de dimensiones y con funcionalidades aplicables de una dimensión a otra.

7) ***Dynamic sparse-matrix handling*:** la manipulación de datos en los sistemas OLAP debe poder diferenciar valores vacíos de valores nulos y además poder ignorar las celdas sin datos.

8) **Soporte multiusuario:** que significa que puede dar respuesta a diferentes necesidades de negocio, y que cada usuario puede tener acceso a un determinado subconjunto de datos.

9) **Operaciones cruzadas entre dimensiones sin restricciones:** todas las dimensiones son creadas igual y las operaciones entre dimensiones no deben restringir las relaciones entre celdas.

10) **Manipulación de datos intuitiva:** dado que los usuarios a los que se destinan este tipo de sistemas son frecuentemente analistas y altos ejecutivos, la interacción debe considerarse desde el prisma de la máxima usabilidad de los usuarios.

11) ***Reporting flexible*:** los usuarios deben ser capaces de manipular los resultados que se ajusten a sus necesidades conformando informes. Además, cambios en el modelo de datos deben reflejarse automáticamente en esos informes.

12) **Niveles de dimensiones y de agregación ilimitados:** no deben existir restricciones para construir cubos OLAP con dimensiones y con niveles de agregación ilimitados.

2. OLAP en el contexto de Pentaho

Mondrian es el motor OLAP integrado en Pentaho y ha sido renombrado como *Pentaho Analysis Services*. Este motor se combina con un visor OLAP que es diferente si consideramos la versión *community* o *enterprise* y con dos herramientas de desarrollo.

Resumiendo:

Tabla 1

	Community	Enterprise
Motor OLAP	Mondrian	
Visor OLAP	Jpivot, Saiku Analytics, Pivot4j	Pentaho Analyser (anteriormente Clearview), Saiku Analytics Enterprise
Herramientas de desarrollo	Pentaho Schema Workbench, Pentaho Aggregation Designer	PDI Enterprise

Trataremos cada uno de esos puntos en los siguientes apartados.

2.1. Mondrian

Mondrian es un servidor/motor OLAP escrito en java que está licenciado bajo la *eclipse public license* (EPL). Como proyecto existe desde 2003 y fue adquirido por Pentaho en 2005.

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 están en una base de datos en la que existen las tablas que conforman la información multidimensional con la que el motor trabaja. El lenguaje de consulta es MDX.

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

Esta forma de trabajar tiene ciertas ventajas:

- No se generan cubos/estructuras OLAP estáticas, y por lo tanto se ahorra el tiempo de generación y el espacio.

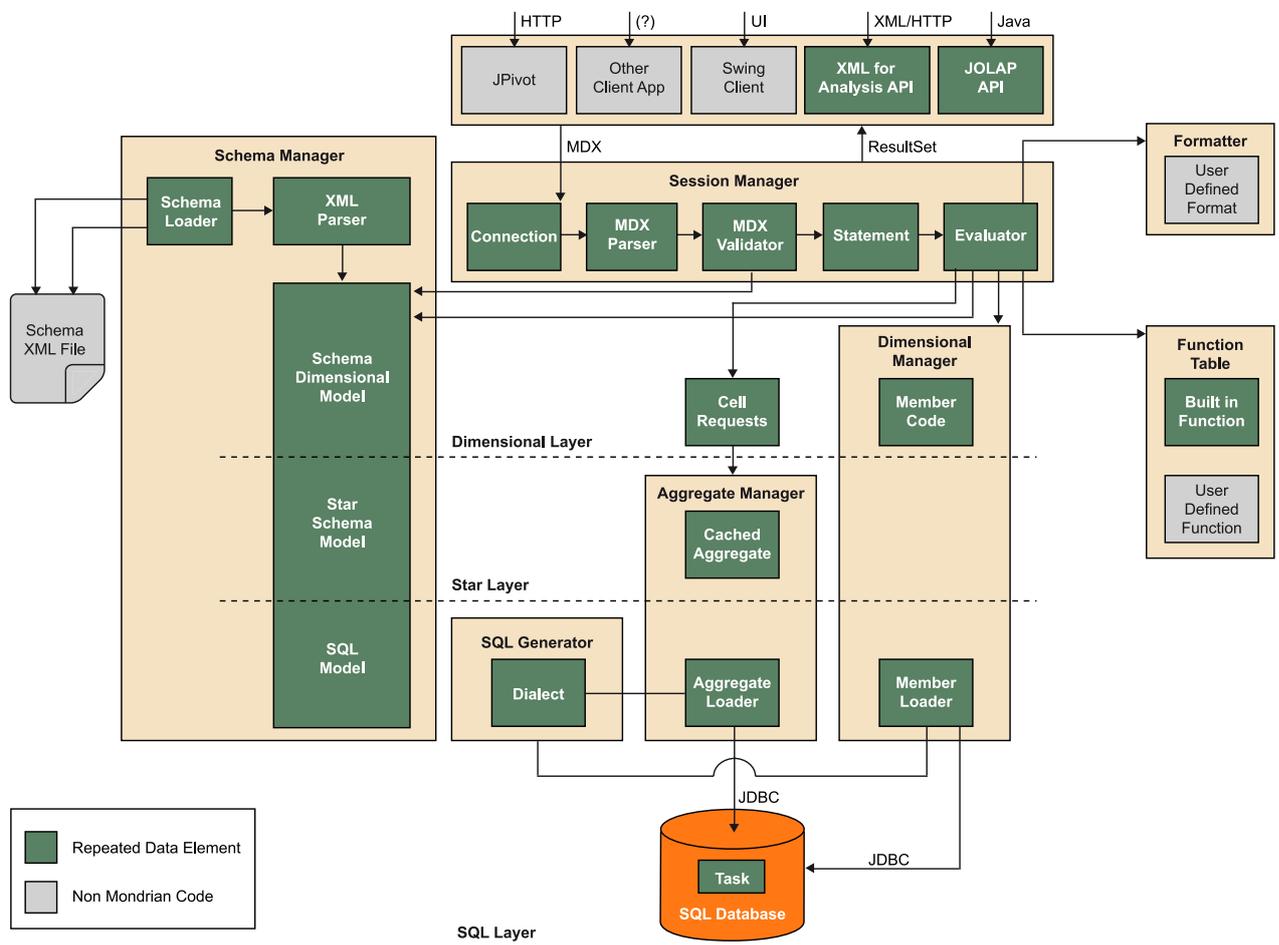
- Se trabaja con datos actualizados siempre al utilizar la información residente en la base de datos.
- Mediante el uso del caché y de tablas agregadas, se pretende simular el mejor rendimiento de los sistemas MOLAP.

Pero también presenta varias desventajas:

- En el caso de apagar el servidor de Pentaho, la memoria caché se vaciará.
- No existe la posibilidad de realizar consultas *off-line*.

El siguiente diagrama presenta la arquitectura de Mondrian:

Figura 2. Arquitectura de Mondrian



Fuente: Pentaho.

Mondrian funciona sobre las bases de datos estándar del mercado: Oracle, DB2, SQL Server, MySQL, PostgreSQL, LucidDB, Teradata, etc., lo que habilita y facilita el desarrollo de negocio.

Los últimos desarrollos de Mondrian se caracterizan por incluir `olap4j`. Es una iniciativa del mismo desarrollador de Mondrian: Julian Hyde.

Mondrian es un caso atípico en el contexto *open source business intelligence* (OSBI). Para la gran mayoría de las herramientas de inteligencia de negocio existe una o varias opciones. En el caso de soluciones ROLAP, Mondrian es el único producto.

Varios fabricantes han incluido Mondrian en sus soluciones (Tibco JasperSoft, OpenI, SpagoBI e incluso los proyectos que ya no están en desarrollo o han sido comprados, como Lucidera y OpenReports). El hecho de existir una única solución y toda una comunidad de fabricantes y usuarios a su alrededor genera que el equipo de desarrollo de Mondrian (dirigido por Julian Hyde) tenga ciclos de desarrollo mucho menores que otras soluciones.

La versión actual es la 4, cuyo desarrollo se inició en 2014. Sin embargo, la que usa Pentaho es la versión 3, por lo que lo explicado se aplica a dicha versión. La versión 4 introduce mejoras en el esquema de definición del cubo, dimensiones, jerarquías, atributos y métricas. Los cambios son suficientemente relevantes, por lo que supone un rediseño de los visores OLAP y de las herramientas de diseño que aún está en proceso, y no existe fecha para la versión estable y definitiva.

2.2. Visores OLAP

Como ya se ha comentado, en el contexto de Pentaho, existen cuatro visores OLAP: JPivot, Saiku Analytics, Pivot4j y Pentaho Analyser.

1) JPivot

JPivot es un cliente OLAP basado en JSP que empezó en 2003 y que desde 2013 no recibe actualizaciones. Tradicionalmente se ha considerado un proyecto hermano de Mondrian, dado que combinado con él permite realizar consultas tanto MDX como a partir de elementos gráficos que se muestran en un navegador web. Durante largo tiempo ha sido el único visor existente para Mondrian. Pentaho adaptó su estilo para diferenciarlo de la interfaz original.

Las características principales de este visor analítico son las siguientes:

- Capacidades de análisis interactivo a través de un acceso web basado en Excel, lo que le proporciona una alta funcionalidad.
- Está basado en estándares de la industria (JBDC, JNDI, SQL, XML/A y MDX).
- Posibilidad de extenderse mediante desarrollo.

Olap4j

Es una API java cuyo objetivo es permitir la creación de aplicaciones OLAP intercambiables entre los diferentes motores OLAP del mercado.

Julian Hyde

Actualmente Julian trabaja en HortonWorks, creadores de una de las principales distribuciones de Hadoop.

Figura 3. Interfaz de Jpivot



						Measures		
Region	My Image	P	P0	P1	P2	▲ Measures[0]	▲ Measures[1]	▲ Measures[2]
-All Region[0]		0	V00			856.44 ↗	820.95 ↗	983.55 ↘
+Region[0]		0		V10		1,095.05 ↘	1,087.79	958.23
-Region[1]		1		V11		1,052.05	1,107.59	980.32
+City[0]	✓	0			V20	1,088.31	884.90	1,085.73
+City[1]	✓	1			V21	1,336.43	943.27	1,117.62
+City[2]	✓	2			V22	969.34	1,086.70	1,094.71
+City[3]	✓	3			V23	964.30	900.58	953.65 ↗
+City[4]	✓	4			V24	915.59 ↗	1,032.98 ↘	876.19 ↘
+City[5]	✓	5			V25	1,023.12	1,242.39	988.85
+City[6]	✓	6			V26	949.16	941.62	1,077.48
+City[7]	✓	7			V27	1,055.92	1,095.56	1,078.26
+Region[2]		2		V12		1,027.36	1,084.88	1,009.71
+Region[3]		3		V13		972.10	932.27	836.34
+Region[4]		4		V14		919.44	926.47 ↗	873.20 ↗

Fuente: Josep Curto.

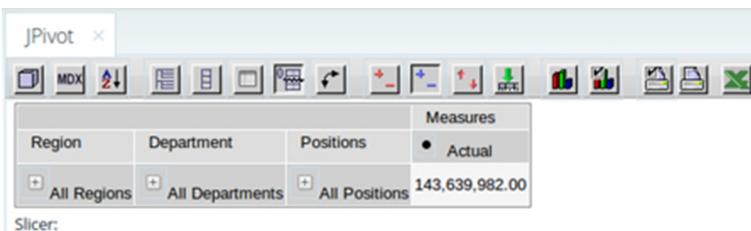
El menú de JPivot ofrece varias opciones al usuario final:

- **Navegador OLAP:** permite determinar las dimensiones que aparecen en las filas y/o columnas, así como los filtros aplicados y las métricas por aparecer.
- **Consulta MDX:** permite visualizar y editar la consulta MDX que genera el informe OLAP.
- **Configurar tabla OLAP:** permite configurar aspectos por defecto de la tabla OLAP, como el orden ascendente o descendente de los elementos.
- **Mostrar miembros padre:** permite mostrar u ocultar el padre del miembro de una jerarquía.
- **Ocultar cabeceras:** muestra u oculta las cabeceras repetidas para facilitar la comprensión del contenido.
- **Mostrar propiedades:** muestra u oculta las propiedades de los miembros de una jerarquía.
- **Borrar filas o columnas vacías:** muestra u oculta los valores sin contenido (conveniente para ciertos informes).
- **Intercambiar ejes:** intercambia filas por columnas.
- **Drill buttons:** *member*, *position* y *replace*:
 - *Member*: permite expandir todas las instancias de un miembro.
 - *Position*: permite expandir la instancia seleccionada de un miembro.
 - *Replace*: permite sustituir un miembro por sus hijos.

- **Drill through:** permite profundizar en el detalle de información a partir de un nivel de información agregado superior.
- **Mostrar gráfico:** muestra un gráfico asociado a los datos. No todos los datos son susceptibles de generar gráficos consistentes.
- **Configuración gráfico:** permite configurar las propiedades del gráfico. Desde el tipo de gráfico hasta propiedades de estilo como tipo de letra, tamaño o color.
- **Configuración de las propiedades de impresión/exportación:** permite configurar las propiedades de la impresión, como el título, la disposición del papel, el tamaño, etc.
- **Exportar a PDF:** permite generar un PDF con el contenido del informe.
- **Exportar a Excel:** permite generar un Excel con el contenido del informe.

En el caso de la integración con Pentaho se añade la capacidad de crear y guardar nuevas vistas analíticas. Cabe comentar que la creación no ofrece muchas capacidades de configuración.

Figura 4. Interfaz de JPivot en Pentaho



Fuente: Josep Curto.

Es necesario indicar que JPivot no es una herramienta muy orientada al usuario. No dispone de funcionalidades *drag & drop* ni tampoco otras funcionalidades como la creación de nuevas métricas o jerarquías, lo que limita el valor para un usuario final.

2) Pentaho Analyser

Con el objetivo de diferenciar la versión *enterprise*, Pentaho adquirió el producto Clearview de Lucidera y lo renombró como Pentaho Analyser, sustituyendo a JPivot.

Figura 5. Interfaz de Pentaho Analyser

Line	Territory	Years		
		2003	2004	2005
Classic Cars		115,011	199,372	97,574
		691,273	1,015,790	384,538
		120,696	42,071	18,835
Motorcycles		587,428	581,043	237,791
		60,789	63,159	65,870
		141,836	204,042	161,260
Planes		16,485	31,959	4,176
		178,109	291,421	55,020
		42,663	67,681	11,082
Ships		154,519	209,128	128,008
		60,556	49,177	-
		90,016	202,942	60,985
Trains		-	35,323	3,070
		172,428	186,992	67,845
		14,156	10,453	8,407
Trucks and Buses		58,238	142,904	48,856
		1,681	8,226	-
		29,538	90,973	17,995
		13,279	-	3,524
		28,304	25,551	15,398
		11,298	80,634	53,735
		228,699	185,421	86,859
		44,498	13,349	-

Fuente: Pentaho.

Es una solución que ofrece:

- Capacidades de *drag & drop*.
- Creación de nuevas medidas calculadas.
- Soporte para soluciones BI SaaS.
- Buscador de objetos.
- Funcionalidades encapsuladas (como, por ejemplo, consultas por jerarquía temporal).

Esta solución está completamente orientada al usuario, lo que facilita su trabajo.

3) Pivot4j

Pivot4j nace como el reemplazo de Jpivot al paralizarse este proyecto. Se caracteriza por:

- Soportar cualquier motor OLAP al estar fundamentado en OLAP4J, lo que significa que puede usarse como aplicación independiente a Pentaho.

- Replicar toda la funcionalidad presente en JPivot.
- Estar orientado al usuario, por lo que permite aplicar todo tipo de funciones del lenguaje MDX de forma visual y mediante *drag & drop*.
- Estar disponible para el servidor de Pentaho mediante el *marketplace*.

Figura 6. Interfaz de Pivot4j

Product	Measures
Classic Cars	4,091,420
Motorcycles	1,274,125
Planes	1,076,757
Ships	748,671
Trains	234,469
Trucks and Buses	1,154,281
Vintage Cars	2,066,226

```

1 SELECT [Measures].[Sales] ON COLUMNS, ([Product].[Classic Cars], [Product].[Motorcycles], [Product].[Planes], [Product].[Ships], [Product].[Trains], [Product].[Trucks and Buses], [Product].[Vintage Cars]) ON ROWS FROM [SteelWheelsSales]

```

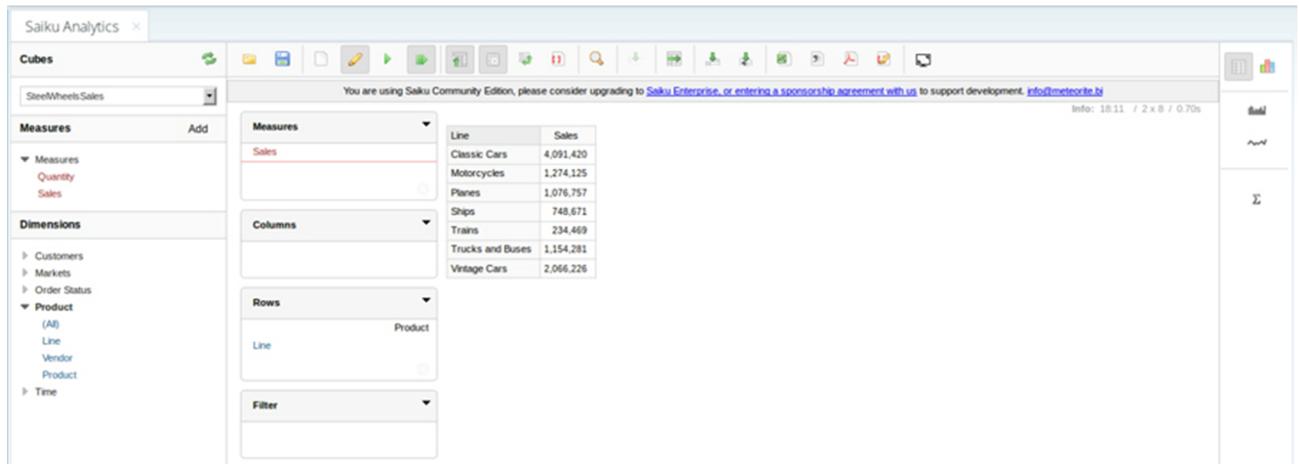
Fuente: Josep Curto.

4) Saiku

Saiku también surge como reemplazo de JPivot, pero en lugar de fundamentarse y replicar JPivot, es un desarrollo completamente nuevo.

- Es un visor OLAP que requiere poca memoria para su ejecución.
- 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. Entre ellas Microsoft SQL Server, Microsoft Analysis Services, Oracle, Oracle Essbase, MongoDB, MySQL, PostgreSQL, Cloudera Impala, Actian Vectorwise, Amazon Redshift, Teradata y Vertica. Aunque no todas están disponibles para la versión *community*.
- Existen tres versiones: *community*, como *plugin* de Pentaho y *enterprise*.
- Está disponible para el servidor de Pentaho mediante el *marketplace*.

Figura 7. Interfaz de Saiku



Fuente: Josep Curto.

2.3. Herramientas de desarrollo

Como ya se ha comentado, en el contexto de Pentaho, existen principalmente tres herramientas de desarrollo: Pentaho Schema Workbench (PSW), Pentaho Aggregation Designer y Pentaho Data Integration (PDI) Enterprise.

En fase de desarrollo y, por lo tanto, no recomendables a nivel de producción, existen en el *marketplace* diversos *plugins* que van a permitir la creación de esquemas OLAP. No vamos a tratarlos en esta asignatura.

1) Pentaho Schema Workbench

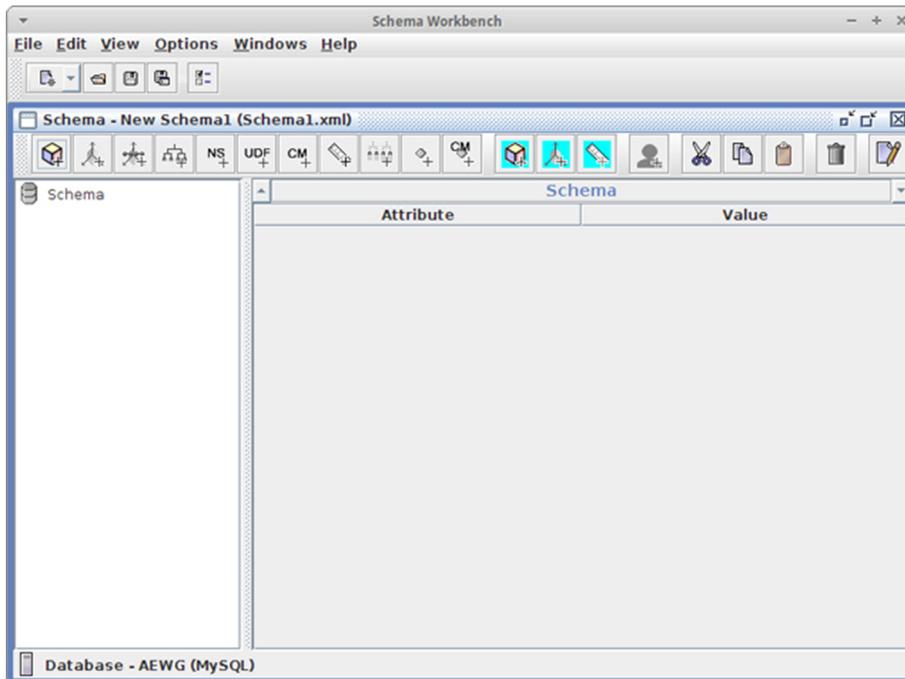
Pentaho Schema Workbench (PSW) es una herramienta de desarrollo que permite crear, modificar y publicar un esquema de Mondrian. Es un programa java multiplataforma.

Es una herramienta muy orientada al desarrollador conocedor de la estructura de un esquema de Mondrian. Permite crear todos los objetos que soporta Mondrian: esquema, cubo, dimensiones, métricas, etc.

Tiene dos áreas: la zona en la que se muestra la estructura jerárquica del esquema OLAP y la zona de edición de las propiedades de cada elemento.

Presenta un menú superior para crear cubos, dimensiones, dimensiones conformadas, métricas, miembros calculados, subconjuntos (*named set*) y crear roles, así como operaciones estándar como cortar, copiar y pegar.

Figura 8. Interfaz de Pentaho Schema Workbench



Fuente: Josep Curto.

Además, entre sus características incluye:

- Capacidad de realizar consultas MDX contra el esquema creado (requiere conocer la sintaxis del lenguaje).
- Consultar la base de datos que sirve de origen para el esquema de Mondrian.
- Publicar directamente el esquema en el servidor de Pentaho.

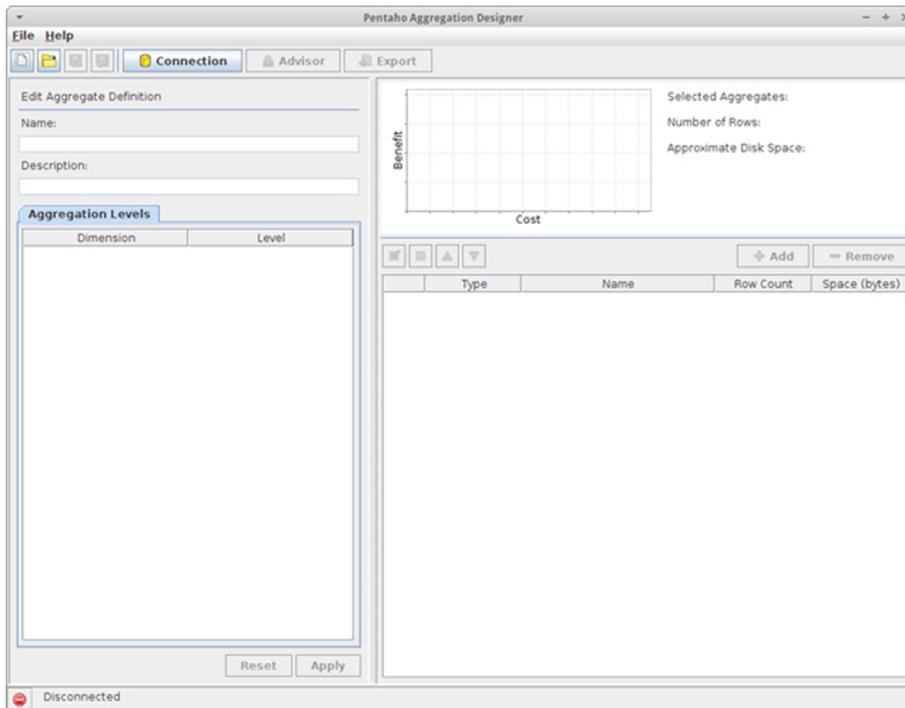
2) Pentaho Aggregation Designer

Un método para optimizar Mondrian, aparte de configurar la caché de este adecuadamente, es a través de la creación de tablas agregadas.

Desde hace pocos meses, Pentaho ofrece una herramienta de diseño orientada a dicha función: Pentaho Aggregation Designer.

Esta herramienta java permite analizar la estructura del esquema de Mondrian contra la cantidad de datos que recuperar y a partir de dicho análisis recomendar la creación de tablas agregadas.

Figura 9. Interfaz de Pentaho Aggregation Designer



Fuente: Josep Curto.

En nuestro caso particular no haremos uso de esta herramienta, aunque se incluye para que pueda ser conocida.

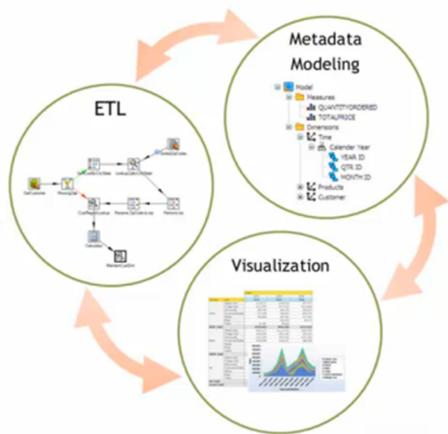
3) PDI Enterprise

En el caso de disponer de la versión *enterprise*, el desarrollador tiene a su alcance otra herramienta para el desarrollo de esquemas OLAP: PDI.

La idea detrás es que después de desarrollar los procesos ETL es el momento adecuado de crear el esquema OLAP para generar valor a los usuarios de negocio. Por ello, PDI Enterprise incluye varias perspectivas de trabajo:

- **Model:** para la creación de esquemas OLAP.
- **Visualize:** para la visualización mediante plantillas de datos en el esquema.
- **Forecast:** para la identificación de tendencias mediante técnicas de series temporales.

Figura 10. Filosofía por perspectivas



Fuente: Pentaho.

El objetivo de las perspectivas es proporcionar un desarrollo ágil de diferentes elementos de la solución de BI en una misma herramienta.

3. Caso práctico

3.1. Diseño OLAP con Pentaho Schema Workbench

El diseño de estructuras OLAP es común en Pentaho y otras soluciones software libre del mercado, dado que las herramientas que proporcionan solo difieren en pequeños puntos de rediseño de la interfaz GUI. El punto realmente diferente es cómo se publican en una u otra plataforma.

En el proceso de creación de una estructura OLAP debemos tener presente que lo que haremos es mapear el diseño de la base de datos (tablas de hecho y dimensiones) con nuestro diseño, de modo que:

- Es posible crear un esquema con menos elementos que los existentes en la base de datos (no interesa contemplar todos los puntos de vista de análisis, por ejemplo).
- Es posible crear un esquema con la misma cantidad de elementos. Se consideran todas las tablas de hecho y las dimensiones.
- Es posible crear un esquema con más elementos que los existentes en la base de datos. Por ejemplo, es posible crear dimensiones u otros objetos que solo existen en el esquema OLAP y que se generan en memoria.

Para este primer ejemplo, vamos a considerar un mapeo uno a uno en el que todos los elementos del *data warehouse* tendrán su correspondencia en la estructura OLAP.

Esta herramienta permite crear elementos en cascada. Es decir, primero creamos un esquema y después tendremos disponibles los elementos del esquema (cubo, dimensión, etc.). Si creamos un cubo, tendremos a nuestra disposición los atributos de un cubo.

También es posible usar el menú que incluye la creación de cubos, dimensiones, jerarquías, niveles, medidas, medidas calculadas, elementos virtuales (cubos, dimensiones y métricas), roles y operaciones estándar, como copiar, cortar, pegar e incluso la edición del XML (que conforma el esquema) de forma directa.

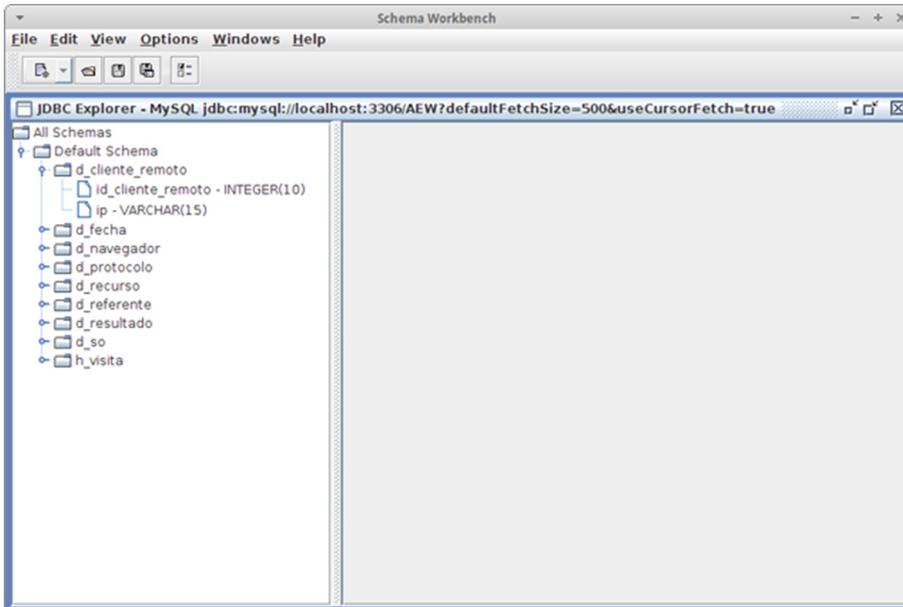
Figura 11. Menú de elementos de un cubo



Fuente: Josep Curto.

Por otra parte, esta herramienta incluye un explorador de la base de datos que, una vez creada la conexión a la base de datos, nos permite explorar la estructura de las tablas para recordar cuál es el nombre de los campos y los atributos que usar.

Figura 12. Explorador de base de datos



Fuente: Josep Curto.

Esta herramienta se inicia a partir del fichero workbench.sh (o workbench.bat en el caso de Windows).

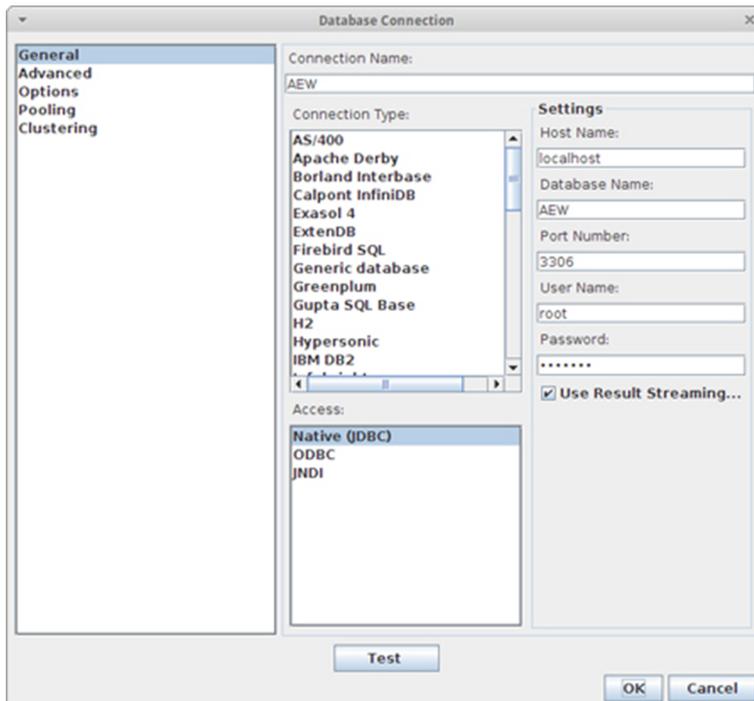
Los pasos en el proceso de creación son los que se exponen a continuación.

Creación de una conexión al *data warehouse*. La herramienta de diseño necesita conocer cuál es la fuente de origen de tablas y datos. Por ello, antes de empezar cualquier diseño es necesario dicha conexión. Una vez creada se guarda en memoria y queda grabada para futuras sesiones. En nuestro caso los parámetros de conexión son:

- *Connection name*: AEW.
- *Connection type*: MySQL.
- *Access*: Native (JDBC).
- *Host name*: localhost.
- *Database name*: AEW.
- *Port number*: 3306.
- *User name*: root.
- *Password*: pentaho.

Es necesario recordar que en el caso de que la base de datos fuera diferente, estos parámetros serían diferentes. En tal caso también sería necesario comprobar la inclusión del *plugin* jdbc en la herramienta.

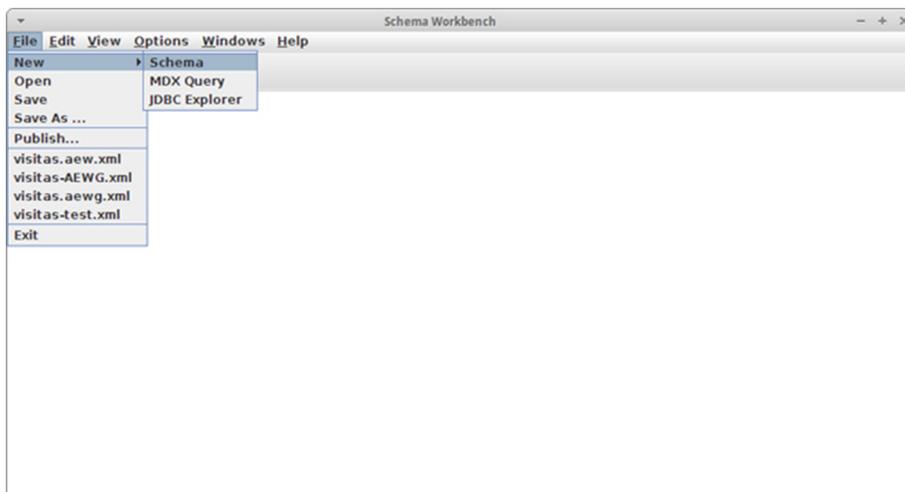
Figura 13. Conexión a la base de datos



Fuente: Josep Curto.

Una vez creada la conexión, podemos crear nuestro primer esquema. Los pasos son: crear un esquema, uno o varios cubos, una o varias tablas de hecho, una o varias dimensiones y una o varias métricas.

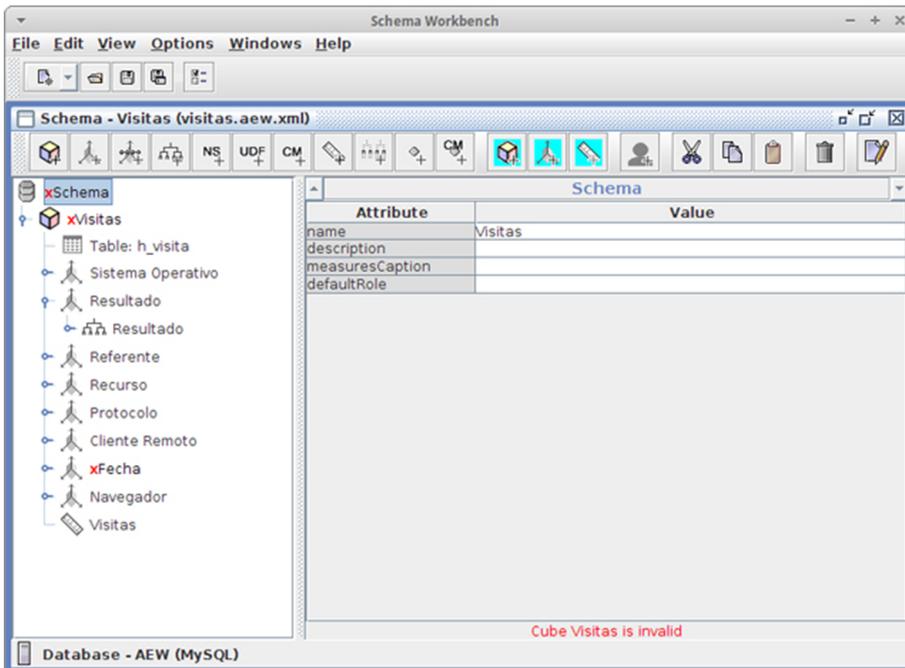
Figura 14. Creación de un nuevo esquema



Fuente: Josep Curto.

Para entender el proceso analizaremos un esquema creado completamente. Primero completamos el esquema introduciendo el nombre del esquema. En nuestro caso, por ejemplo, visitas. En el caso de que se vayan creando esquemas que contiene diversos cubos la recomendación sería nombrarlo con el nombre del proyecto, AEW.

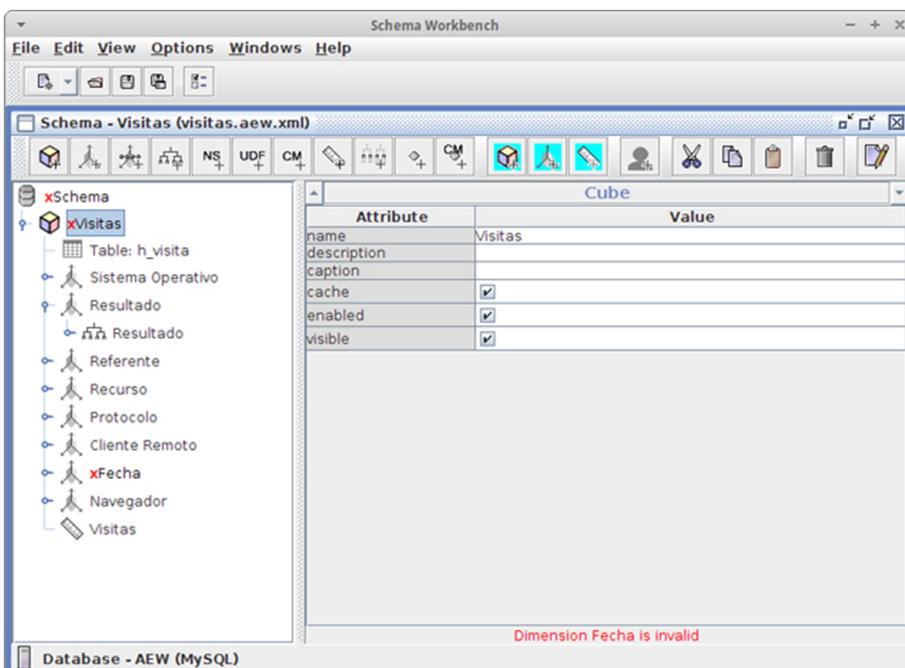
Figura 15. Definición del esquema



Fuente: Josep Curto.

Creamos un cubo. Debemos definir el nombre y activar las opciones de *visible*, *enabled* y *cache*. Esta última opción es importante dado que indica al motor Mondrian que las consultas que se hagan deben guardarse en la caché, lo que habilita el enfoque híbrido.

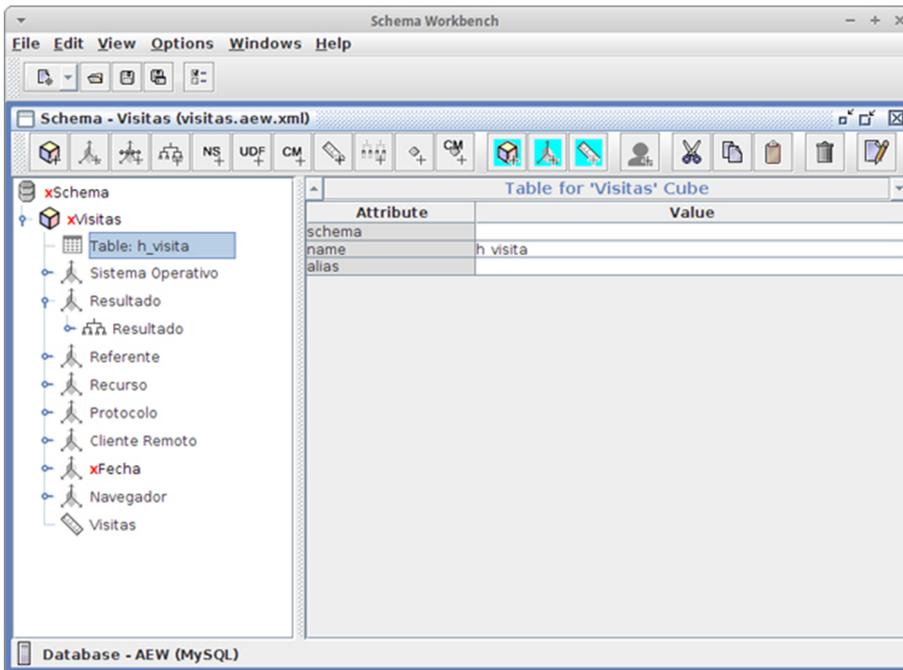
Figura 16. Definición del cubo de un esquema



Fuente: Josep Curto.

Todo cubo necesita una tabla de hecho. En nuestro caso la de visitas. En el campo *name*, elegimos la tabla del *data warehouse* que tiene el rol de tabla de hecho.

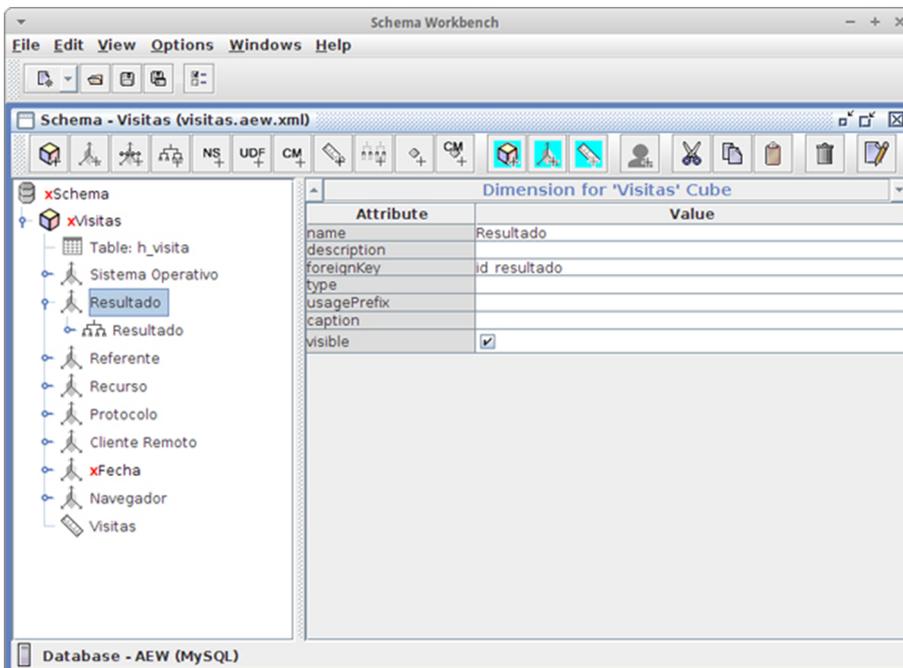
Figura 17. Definición de la tabla de hecho del cubo



Fuente: Josep Curto.

Un cubo necesita al menos de una dimensión. Analizamos el caso de creación de la dimensión resultado. Como mínimo, debemos definir su nombre y la correspondiente clave foránea (*foreign key*).

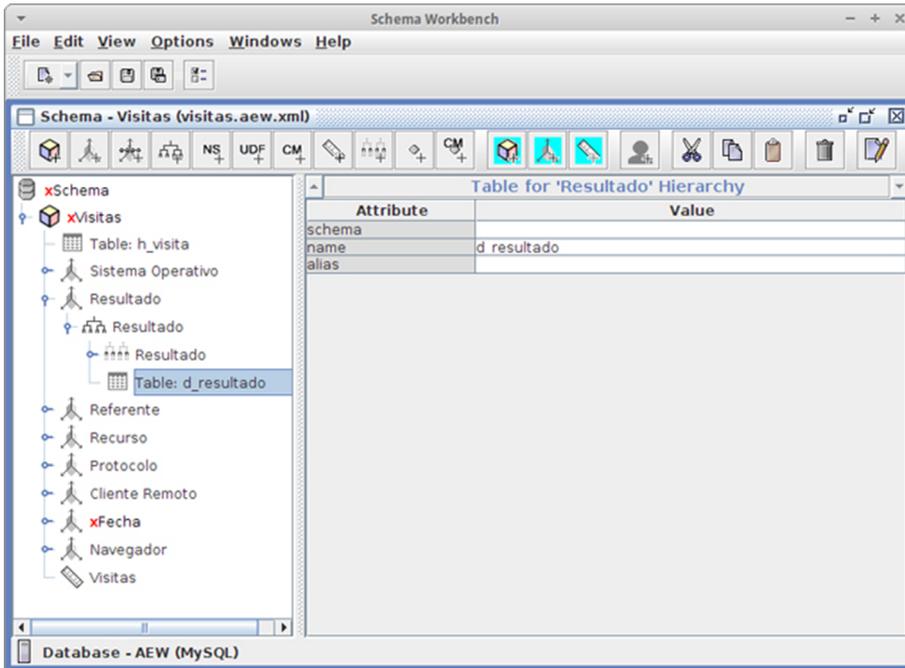
Figura 18. Definición de una dimensión del cubo



Fuente: Josep Curto.

Es necesario definir la tabla de que contiene la información de la dimensión. En nuestro caso, d_resultado.

Figura 19. Definición de la tabla de una dimensión



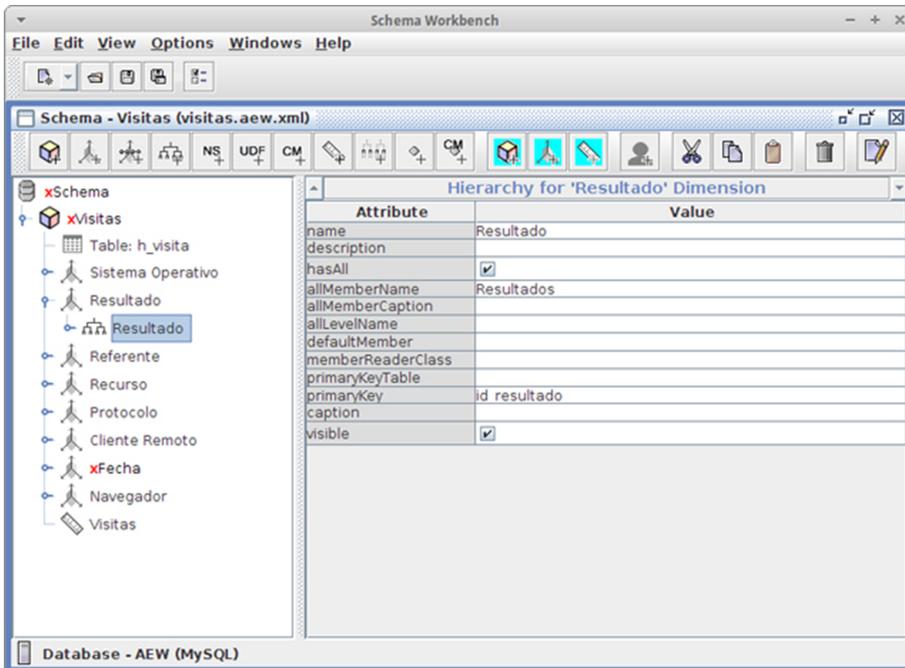
Fuente: Josep Curto.

Toda dimensión tiene una o varias jerarquías. Para definir cada jerarquía es necesario definir su nombre, indicar si acumula los valores (hasAll), el nombre de dicha acumulación y, finalmente, su clave primaria.

Dimensión fecha

Un ejemplo claro de jerarquía con múltiples niveles es la dimensión fecha.

Figura 20. Definición de la jerarquía de una dimensión

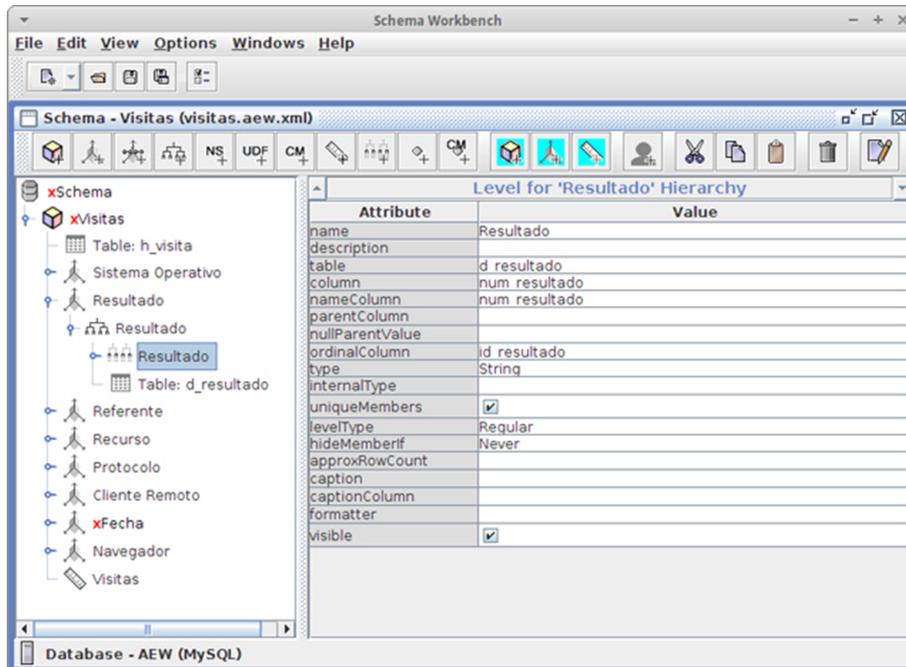


Fuente: Josep Curto.

Toda jerarquía necesita como mínimo un nivel que la componga (y una jerarquía puede tener más de un nivel). Para definir correctamente el nivel, debemos especificar el nombre del nivel, la tabla donde está la información, la

columna que contiene dicha información, cómo se ordenan los resultados, la tipología del valor, la tipología de nivel, si los valores son únicos y si es necesario ocultar el nivel en algún caso.

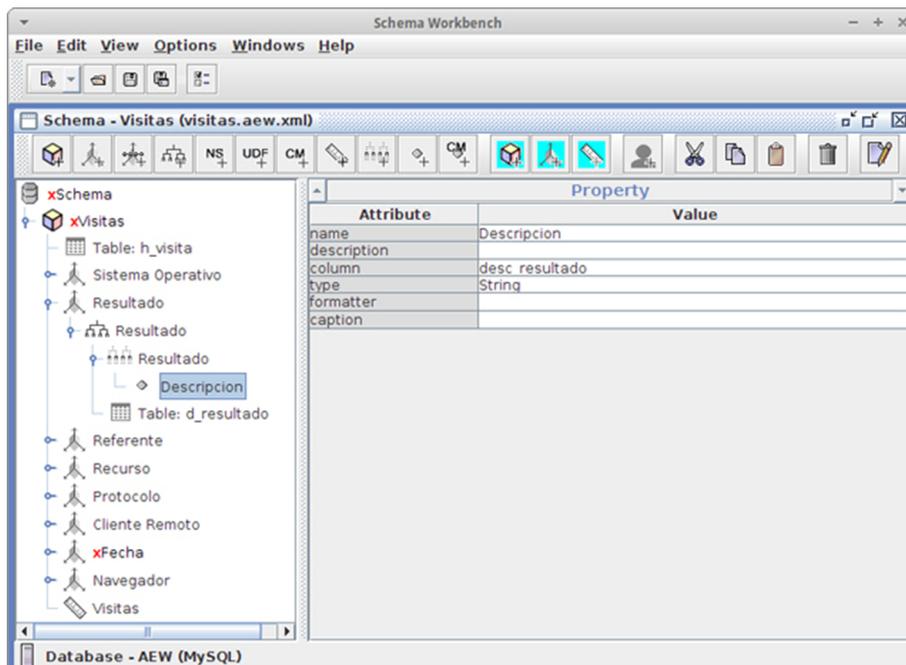
Figura 21. Definición del nivel de una jerarquía



Fuente: Josep Curto.

Algunos miembros contienen información adicional que podemos mostrar como propiedades. En este caso, para resultado, podemos mostrar la descripción del valor. Añadimos una propiedad y definimos el nombre, la columna que contiene la información y la tipología del valor.

Figura 22. Definición de la propiedad de un nivel

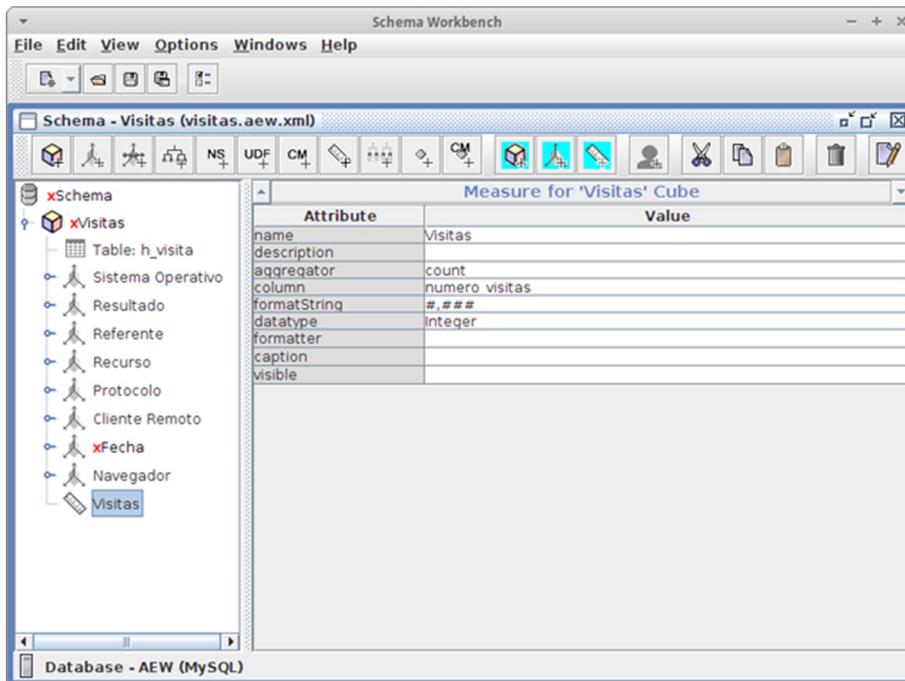


Fuente: Josep Curto.

El resto de las dimensiones se definen de modo similar. Una vez definidas las dimensiones, es necesario definir las métricas. En este caso, solo definimos las visitas. Para definir una métrica debemos cumplimentar varios atributos:

- *name*: nombre de la vista.
- *aggregator*: tipo de agregación, en nuestro caso, *count*.
- *column*: columna (de la base de datos) desde la que se recupera el dato, en nuestro caso *numero_visitas*.
- *formatString*: formato del número.
- *datatype*: tipo de número, en nuestro caso entero.

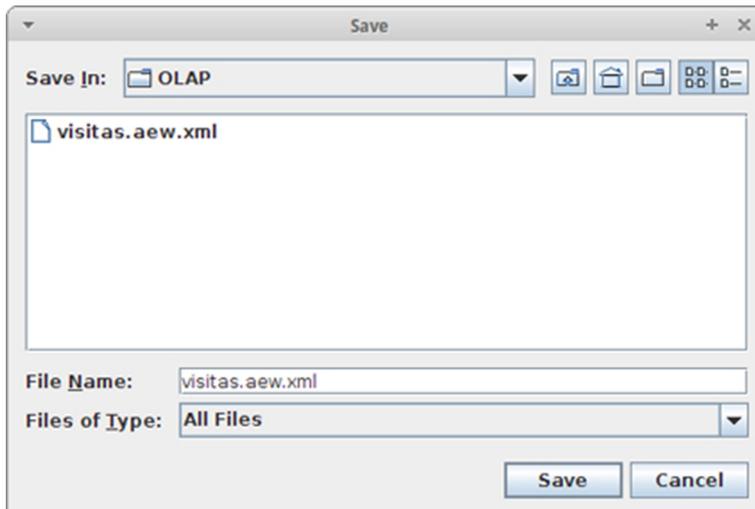
Figura 23. Definición de la métrica visitas en el esquema



Fuente: Josep Curto.

Finalmente, guardamos el esquema creado en la carpeta OLAP que hemos definido anteriormente y lo renombramos como *visita.aew.xml*.

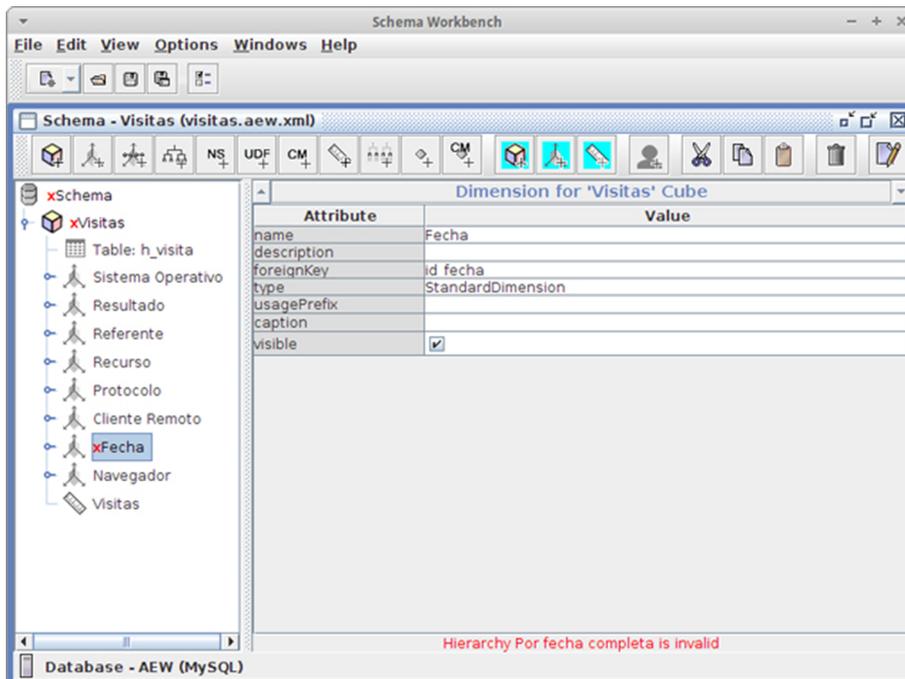
Figura 24. Guardando el esquema OLAP



Fuente: Josep Curto.

La herramienta de desarrollo comprueba que el esquema que vamos creando está bien definido (en caso negativo, en la parte inferior aparece un mensaje en rojo). No obstante, existen algunos escenarios en los que PSW muestra un error pero en realidad es un falso positivo. Esto se debe a que el motor Mondrian soporta funcionalidades no implementadas en PSW.

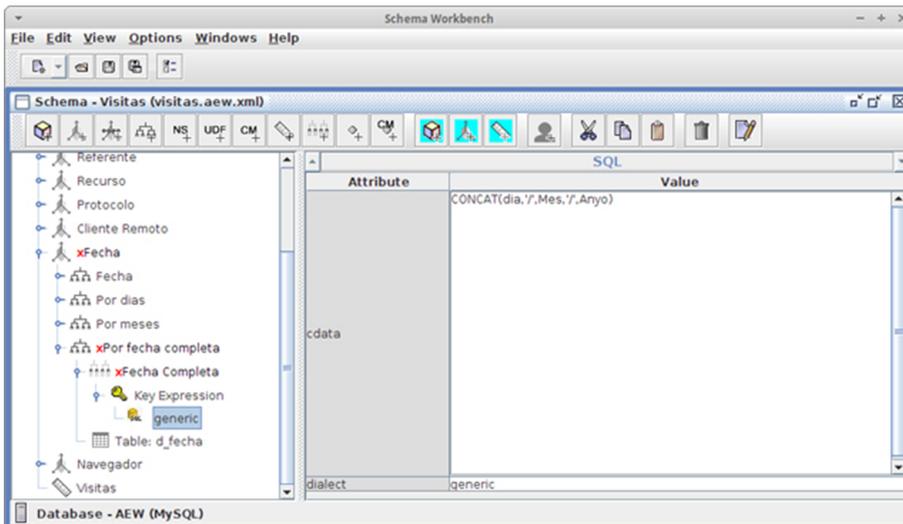
Figura 25. Falso positivo en PSW



Fuente: Josep Curto.

En este caso hemos diseñado una jerarquía para la dimensión fecha a partir de una expresión SQL soportada por el motor OLAP pero no incluida como funcionalidad de PSW.

Figura 26. Uso de expresiones SQL en PSW



Fuente: Josep Curto.

Podemos comprobar que el diseño que hemos realizado funciona correctamente y responde a nuestras preguntas. Para ello, usamos *MDX query*. Si la conexión a base de datos se ha definido correctamente y el esquema OLAP está bien definido, se conectará con éxito. Para poder comprobar el funcionamiento es necesario escribir una consulta MDX. El lenguaje MDX es similar a SQL pero es más complejo. Vamos a considerar una consulta sencilla en la que ponemos las medidas en la columna y la dimensión sistema operativo en las filas. La consulta tiene la siguiente forma:

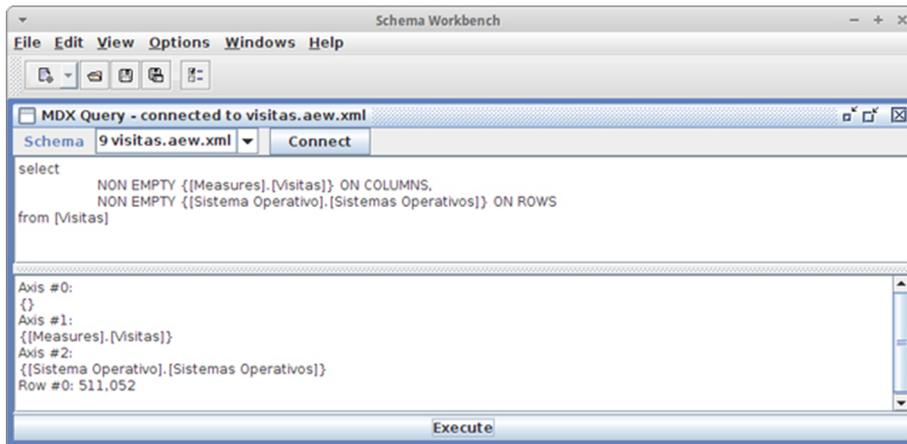
```
SELECT
    NON EMPTY {[Measures].[Visitas]} ON COLUMNS,
    NON EMPTY {[Sistema Operativo].[Sistemas Operativos]} ON ROWS
FROM [Visitas]
```

Lectura recomendada

Consultar el anexo que proporciona una introducción a MDX. Para más información se recomienda la lectura del libro:

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

Figura 27. Consulta MDX mediante PSW



Fuente: Josep Curto.

El resultado son 511.052 visitas.

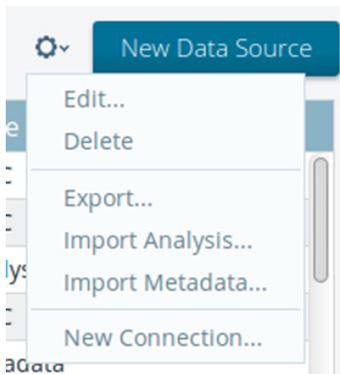
3.2. Publicación de un esquema OLAP en Pentaho Server

La publicación del esquema OLAP en Pentaho Server se puede hacer directamente en Pentaho Schema Workbench o mediante el servidor de Pentaho. Explicaremos la segunda opción.

Para tener disponible el esquema OLAP es necesario hacer un par de cosas. Primero, el servidor debe reconocer la base de datos AEW como fuente de datos de la plataforma. Y la segunda, la publicación del esquema debe estar vinculada a esta fuente de datos.

En el menú de entrada pulsamos *Manage Data Sources*. Se abrirá una nueva ventana emergente. Aparecerá la lista de las existentes y su tipo (como por ejemplo *Sample Data*) y pulsamos las propiedades.

Figura 28. Opciones disponibles para las fuentes de datos



Fuente: Josep Curto.

Pulsamos en *New Connection* y se abrirá un menú de configuración de bases de datos ya conocido (es el mismo que se emplea en Pentaho Data Integration; este tipo de coherencia en el diseño de la solución facilita la implementación y el uso).

Seleccionamos MySQL y debemos configurar nombre de la conexión, *host*, nombre de la base de datos, puerto, usuario y contraseña.

Figura 29. Definición de una nueva base de datos

The screenshot shows the 'Database Connection' dialog box with the following configuration:

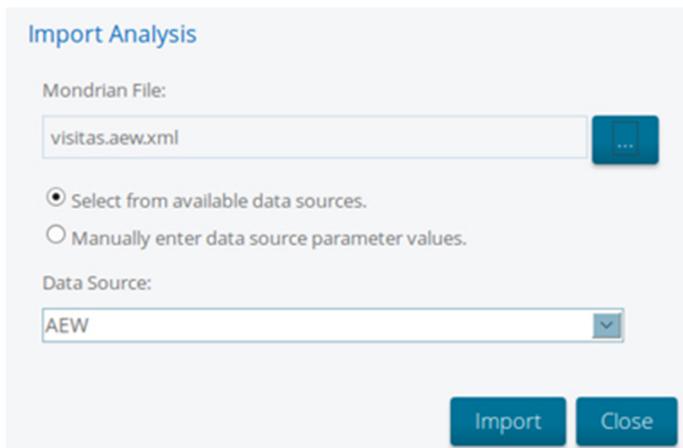
- Connection Name:** AEW
- Database Type:** MySQL
- Settings:**
 - Host Name: localhost
 - Database Name: AEW
 - Port Number: 3306
 - User Name: root
 - Password: (empty)
- Access:** Native (JDBC)

Buttons: Test, OK, Cancel

Fuente: Josep Curto.

Una vez configurada, la base de datos AEW estará disponible. Ahora ya podemos importar al sistema el esquema, puesto que necesita poder conectarse contra la base de datos correcta. De nuevo vamos a las propiedades (*settings*) y seleccionamos *Import Analysis*. Se abrirá un nuevo menú y debemos seleccionar el fichero *visitas.aew.xml*. También debemos indicar cuál es la fuente de datos. En nuestro caso AEW. Pulsamos *Import*.

Figura 30. Publicación de un nuevo esquema



Import Analysis

Mondrian File:

visitas.aew.xml

Select from available data sources.

Manually enter data source parameter values.

Data Source:

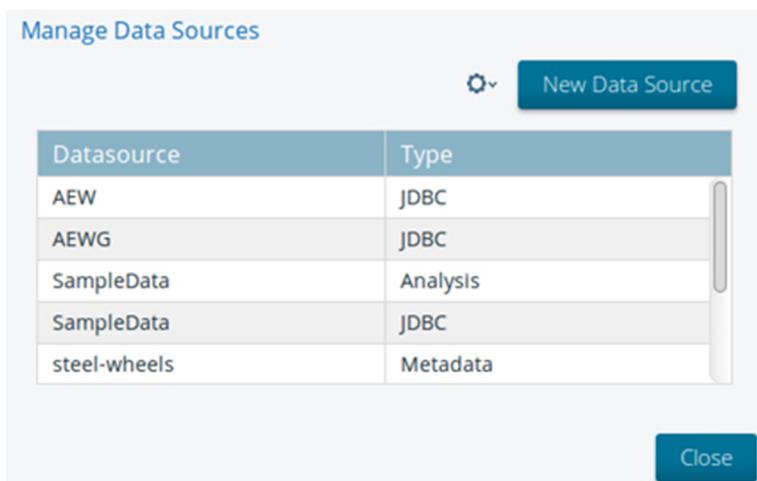
AEW

Import Close

Fuente: Josep Curto.

De este modo tendremos disponibles las diferentes fuentes de datos disponibles.

Figura 31. Fuentes disponibles



Manage Data Sources

New Data Source

Datasource	Type
AEW	JDBC
AEWG	JDBC
SampleData	Analysis
SampleData	JDBC
steel-wheels	Metadata

Close

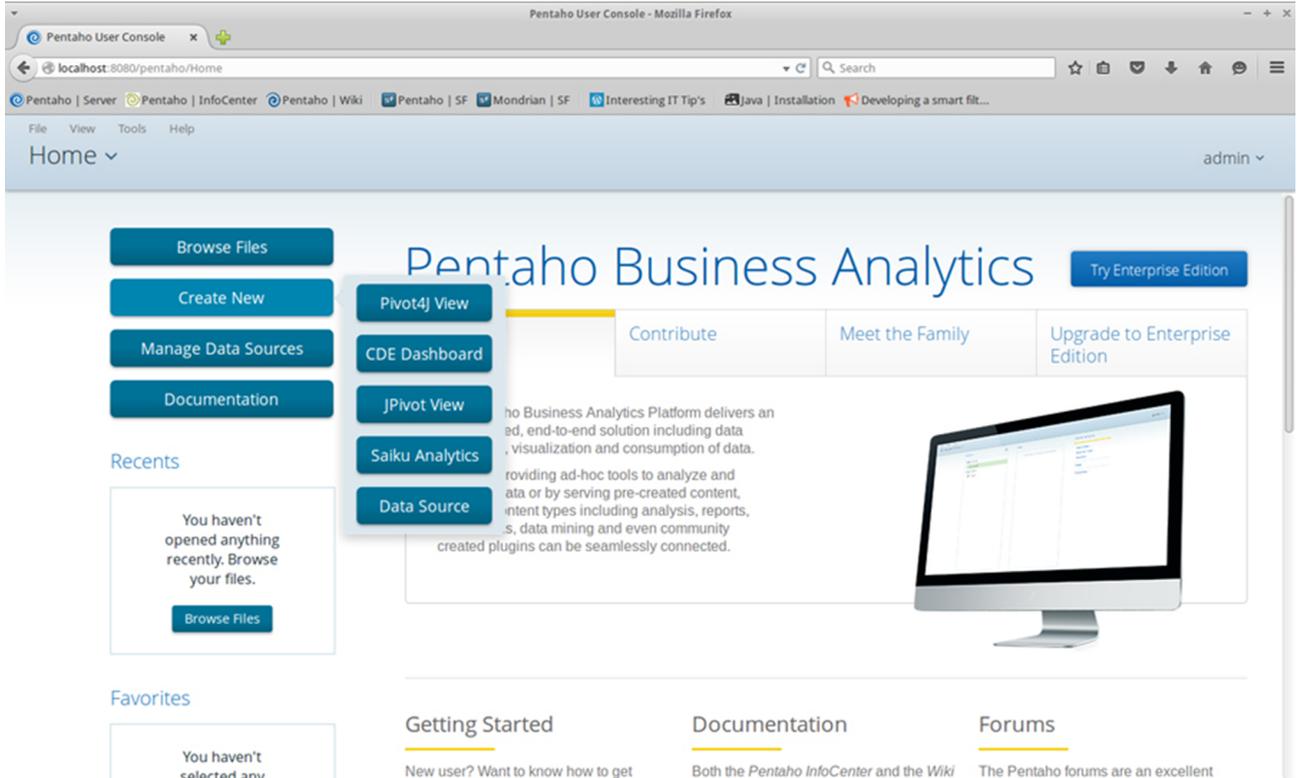
Fuente: Josep Curto.

Finalmente, podemos crear una nueva consulta a través de la interfaz web de Pentaho. En la pantalla inicial seleccionamos *Create New*. Tendremos tantas opciones como visores instalados en el servidor. En nuestro caso tenemos los tres *open source*. Seleccionamos *Pivot4j View*.

Nota

En el caso de disponer solo de Jpivot, es necesario instalar el resto a través del Pentaho *marketplace*. Es posible encontrarlo pulsando en *Home* y luego la opción *marketplace*. Se abrirá una nueva ventana con los *plugins* disponibles.

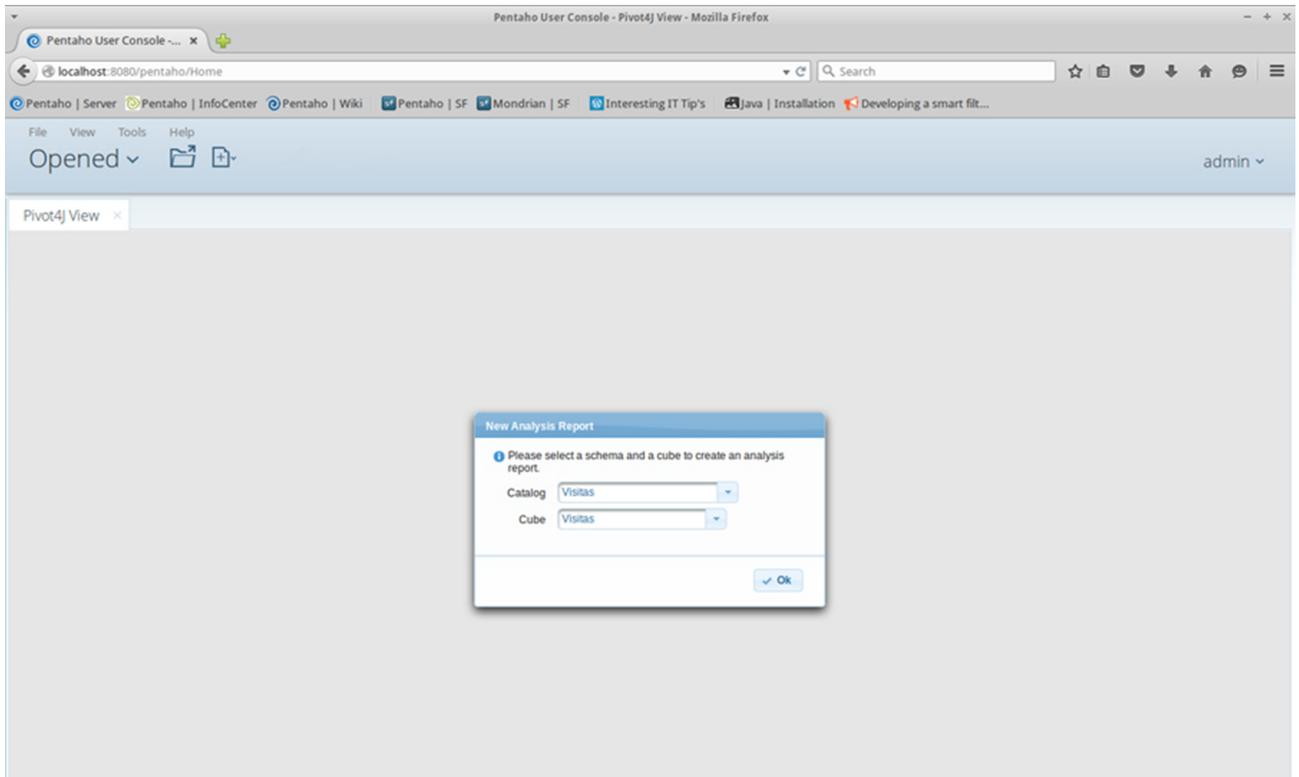
Figura 32. Selección visor OLAP



Fuente: Josep Curto.

Elegimos el esquema y el cubo visitas que es el nuestro.

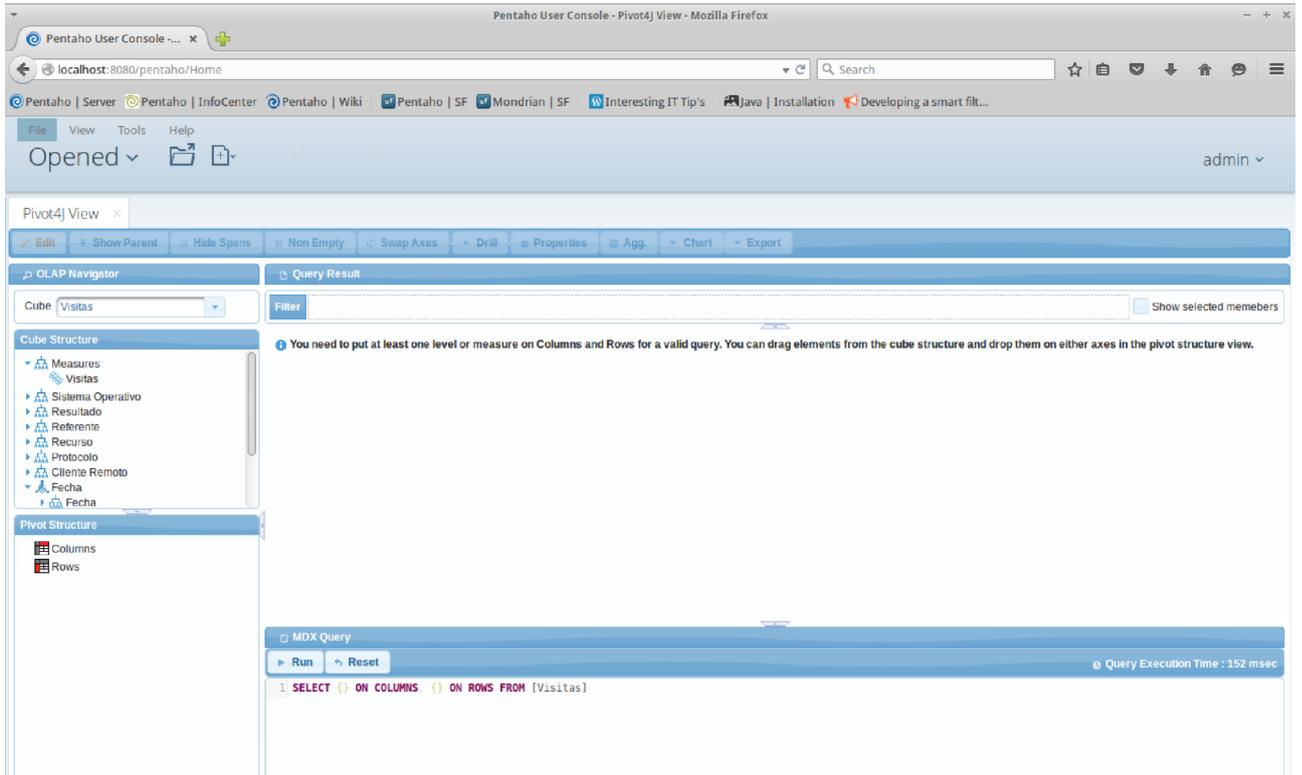
Figura 33. Selección esquema OLAP en Pivot4j



Fuente: Josep Curto.

Aparecerá la interfaz de Pivot4j, que, por defecto, está vacía.

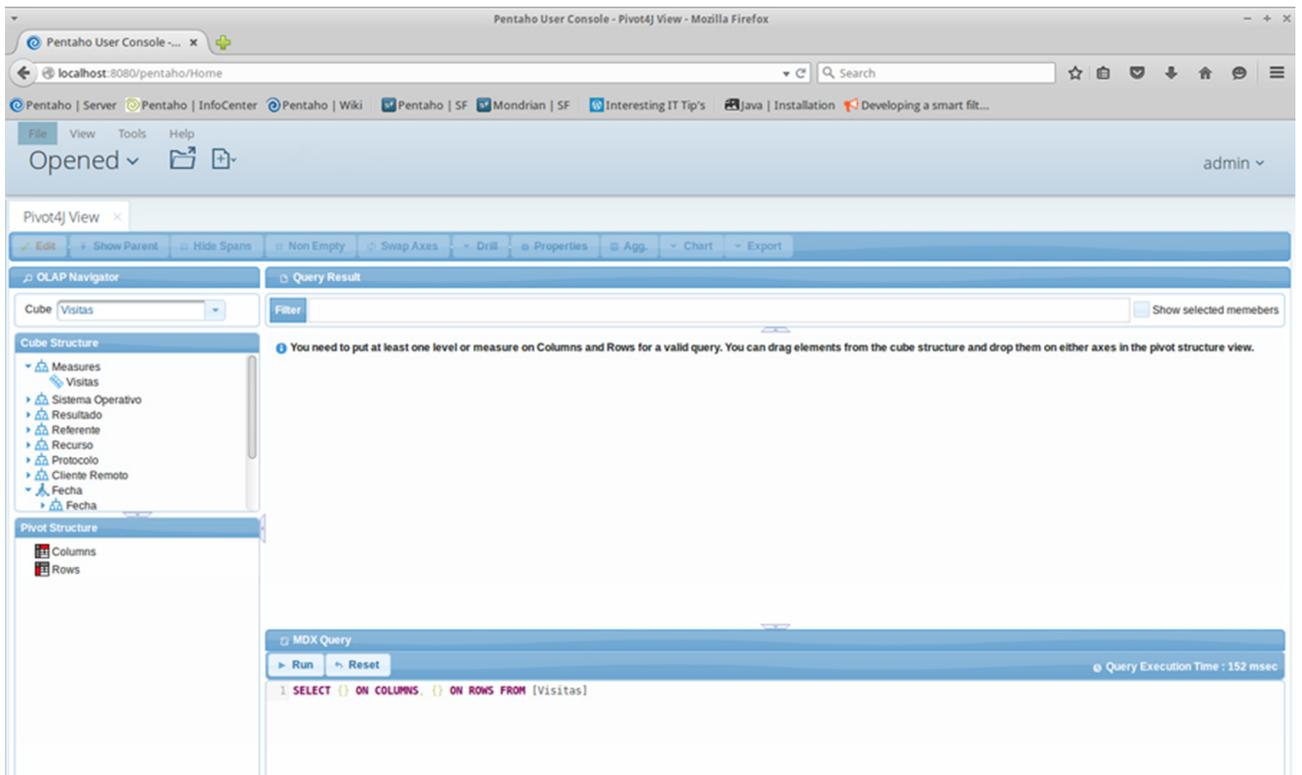
Figura 34. Interfaz de Pivot4j



Fuente: Josep Curto.

Mediante *drag & drop*, seleccionamos la medida visitas en las columnas y el detalle de los resultados en las filas. Automáticamente se calcula el resultado y se genera la consulta MDX.

Figura 35. Consulta OLAP en Pivot4j



Fuente: Josep Curto.

Abreviaturas

DOLAP *Desktop online analytical processing.*

EPL *Eclipse public license.*

HOLAP *Hybrid online analytical processing.*

JDBC *Java database connectivity.*

JNDI *Java naming and directory interface.*

JSP *Java server page.*

MDX *Multidimensional expressions.*

MOLAP *Multidimensional online analytical processing.*

MTD *Month to day.*

OLAP *Online analytical processing.*

PAT *Pentaho analysis tool.*

PSW *Pentaho schema workbench.*

QTY *Quarter to day.*

ROLAP *Relational online analytical processing.*

SaaS *Software as a service.*

SQL *Structured query language.*

XML/A *XML for analysis.*

WTD *Week to day.*

YTD *Year to day.*

Bibliografía

Bouman, R.; Van Dongen, J. (2009). *Pentaho® Solutions: Business Intelligence and Data Warehousing with Pentaho® and MySQL*. Indianápolis: Wiley Publishing.

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

Schrader, M. y otros (2009). *Oracle Essbase & Oracle OLAP: The Guide to Oracle's Multidimensional Solutions*. Nueva York: McGraw-Hill.

Thomsen, E. (2002). *OLAP Solutions: Building Multidimensional Information Systems*. Hoboken: John Wiley & Sons.

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.

Anexo

MDX

MDX es un lenguaje de consultas OLAP creado en 1997 por Microsoft. No es un estándar pero varios fabricantes lo han adoptado como el estándar de hecho.

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

Web de interés

Para consultar información relativa a MDX y Mondrian, podéis hacerlo en este enlace.

Sintaxis de MDX

La sintaxis de MDX es compleja. La mejor manera de ejemplificarla es a través de un caso determinado. Imaginemos un cubo de ventas con el importe de las ventas y unidades vendidas como medidas y con las siguientes dimensiones:

- Tiempo: temporal de las ventas con niveles de año y mes.
- Familia: productos vendidos con niveles de familia de productos y productos.

Para obtener por ejemplo el importe de las ventas para el año 2016 para la familia de productos lácteos, la consulta sería:

```
SELECT
    { [Measures].[importe ventas] } on columns,
    { [Tiempo].[2016] } on rows
FROM [cubo ventas]
WHERE ([Familia].[lacteos])
```

Es posible observar que la estructura general de la consulta es de la forma SELECT... FROM... WHERE...:

- En el *select* se especifica el conjunto de elementos que queremos visualizar, indicando también qué valores pertenecen a las filas y cuáles a las columnas.
- 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.

Funciones de MDX

MDX incluye múltiples funciones para realizar consultas a través de la jerarquía existente en el esquema OLAP. Podemos destacar entre ellas:

- **CurrentMember**: permite acceder al miembro actual.
- **Children**: permite acceder a todos los hijos de una jerarquía.
- **prevMember**: permite acceder al miembro anterior de la dimensión.
- **CrossJoin** (conjunto_a,conjunto_b): obtiene el producto cartesiano de dos conjuntos de datos.
- **BottomCount** (conjunto_datos,N): obtiene un número determinado de elementos de un conjunto, empezando por abajo, opcionalmente ordenado.
- **BottomSum** (conjunto_datos,N,S): obtiene de un conjunto ordenado los N elementos cuyo total es como mínimo el especificado (S).
- **Except** (conjunto_a,conjunto_b): obtiene la diferencia entre dos conjuntos.
- **AVG**, **COUNT**, **VARIANCE** y todas las funciones trigonométricas (seno, coseno, tangente, etc.).
- **PeriodsToDate**: devuelve un conjunto de miembros del mismo nivel que un miembro determinado, 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.
- **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:** devuelve un miembro de un periodo anterior en la misma posición relativa que el miembro especificado.

Miembros calculados en MDX

Una de las funcionalidades más potentes que ofrece el lenguaje MDX es la posibilidad de realizar cálculos complejos tanto dinámicos (en función de los datos que se están analizando en ese momento) como 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*, *distinctcount*, *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 semafóricas pasando por operadores de condiciones.

Para obtener por ejemplo el beneficio (como resultado de restar los costes a los ingresos) para el año 2016 para todos los productos, la consulta sería:

```
WITH MEMBER [Measures].[Beneficios] AS '[Measures].[Ingresos]-[Measures].[Costes]',
    FORMAT_STRING = '$#,###'
SELECT {[Measures].[Ingresos], [Measures].[Costes]} ON COLUMNS,
    {[Producto].Children} ON ROWS
FROM [Ventas]
WHERE [Tiempo].[2016]
```

