Proyecto Taller V

Toolkit para migración de datos y metadata de estructuras multidimensionales

Migrador OLAP

Gonzalo Harreguy – Álvaro Morales

Supervisores
Gustavo Larriera (Infocorp) – Raúl Ruggia (INCO)

Facultad de Ingeniería, diciembre de 2000
Universidad de la República Oriental del Uruguay

Tabla de contenido

Capítulo 5: Implementación

Capítulo 6: Conclusiones

Bibliografía

Apéndice A: OLAP Services

Apéndice B: Lenguaje Extensible de Marcas (XML)

Apéndice C: Manual de usuario

CAPÍTULO 1

Introducción

1. Introducción

1.1 Panorama general

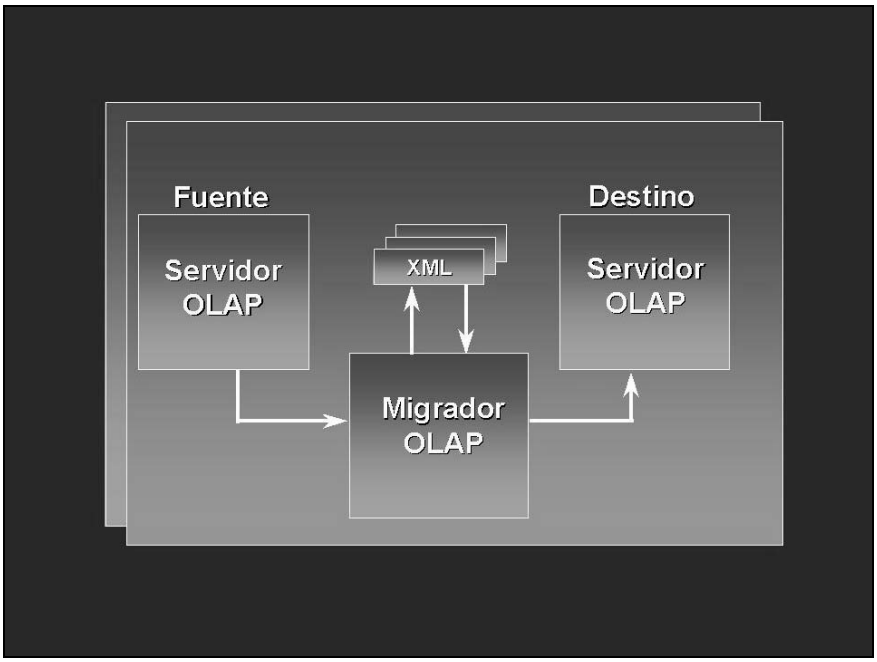
En los últimos años el área de aplicaciones OLAP ha tenido un importante desarrollo. Este tipo de aplicaciones utiliza estructuras multidimensionales para proporcionar un acceso rápido a los datos con el fin de analizarlos. Los datos de origen de OLAP se almacenan habitualmente en almacenes de datos en una base de datos relacional.

A pesar de su amplia difusión, estos productos no se basan en un modelo de datos bien definido ampliamente aceptado, sino que existen un conjunto de funcionalidades que todos proveen, y otras que marcan las diferencias entre los distintos productos.

La mencionada ausencia de un estándar es lo que dificulta el intercambio de información entre distintos servidores OLAP.

Este proyecto pretende migrar la estructura multidimensional utilizada por Microsoft OLAP Services hacia una representación de la misma expresada en XML. Esta representación permitirá reconstruir un cubo en otro servidor OLAP.

El siguiente diagrama ilustra el proceso de migración:



1.2 Objetivos

El propósito del presente proyecto es el de satisfacer los siguientes objetivos:

- Permitir reconstruir un cubo, que se encuentra originalmente en el formato utilizado por Microsoft OLAP Services, en una representación en XML. Este nuevo formato le otorgará una mayor portabilidad a los datos representados mediante la estructura multidimensional.

- Otorgar la facilidad de visualizar la estructura de un cubo de Microsoft OLAP Services en un browser. Para tal fin, la representación de la misma será llevada a HTML.

Para alcanzar los objetivos planteados, el programa desarrollado cuenta con una interfaz gráfica, desde la cual se puede apreciar la metadata del cubo sobre el cuál se está trabajando. Permitiendo realizar distintas operaciones sobre el mismo, ya sean las de exportación, importación y modificación del cubo.

Las operaciones de modificación se podrán realizar tanto sobre el cubo que se obtiene desde el repositorio de OLAP Services, como sobre un cubo que ha sido exportado (encontrándose éste representado en XML).

Los datos de origen de los cubos no serán exportados a la nueva representación (la cual solo expresará la metadata), sino que estos serán transportados directamente de un servidor a otro cuando se realice la importación.

Para la comunicación con Microsoft OLAP Services es utilizado el modelo de objetos de estructuras multidimensionales DSO *(Decisión Support Objects)*, proporcionado por el propio OLAP Services.

1.3 Organización del informe

El informe está compuesto por seis capítulos y tres apéndices. Los capítulos en conjunto pretenden otorgar un panorama acerca del producto desarrollado, así como una reseña sobre los resultados obtenidos en las distintas etapas en las que nos vimos involucrados para su desarrollo. En los apéndices se realiza una descripción de las principales herramientas con las que se debió interactuar, y se incluye un manual de usuario de la aplicación.

- **Capítulo 1: Introducción**
  Se tratan aspectos generales del proyecto y sus objetivos.

- **Capítulo 2: Conocimientos existentes**
  Se habla sobre temas relacionados al área donde se encuentra embebido el proyecto.

- **Capítulo 3: Análisis**
  Análisis general del proyecto, alcance del producto y funcionalidades.

- **Capítulo 4: Diseño**
  Diseño general del producto, así como una visión global acerca de la arquitectura del mismo.

- **Capítulo 5: Implementación**
  Descripción de la etapa de implementación, detalle sobre la interacción llevada a cabo con las herramientas utilizadas.

- **Capítulo 6: Conclusiones y temas pendientes**
  Conclusiones del proyecto y elementos que no fueron posibles incluir en la versión presentada del producto.

- **Apéndice A: Microsoft OLAP Services**
  Descripción de la citada herramienta y sus componentes.

- **Apéndice B: XML**

  Reseña sobre el Lenguaje de Marcado Extensible (XML) y ventajas de su utilización.

- **Apéndice C: Manual de usuario**

  Detalle acerca de la forma de utilización del producto desarrollado.

CAPÍTULO 2

Conocimiento existente

2. Conocimiento existente

2.1 Introducción

A razón de que el presente proyecto consiste en la migración de estructuras multidimensionales, en este capítulo se proporciona un breve resumen acerca de esta tecnología.

Aquí se describen los conceptos básicos de OLAP, así como los elementos fundamentales de los modelos multidimensionales. Finalmente se hace mención de las ventajas que proporciona el uso de la tecnología OLAP en los sistemas de apoyo a la toma de decisiones.

2.2 Proceso analítico en línea (OLAP)

2.2.1 Definición

Tecnología que utiliza estructuras multidimensionales para proporcionar un acceso rápido a los datos con el fin de analizarlos. Los datos de origen de OLAP se almacenan habitualmente en almacenes de datos en una base de datos relacional.

El elemento principal de OLAP es el cubo, que contiene los datos de interés para los usuarios. Los cubos organizan los datos mediante dimensiones y medidas en una estructura multidimensional capaz de responder a las preguntas que los usuarios deseen plantear acerca de los datos de organización.

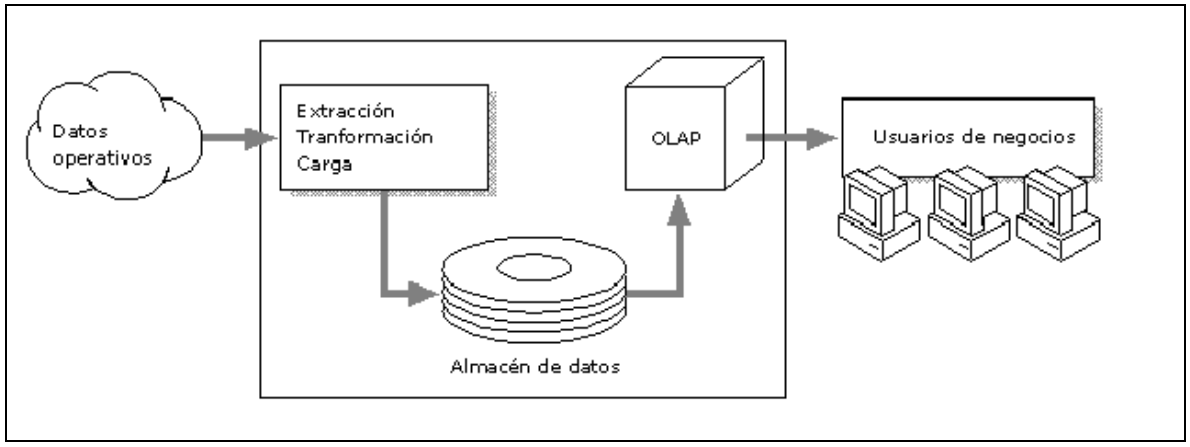
2.2.2 Almacenes de datos

Un almacén de datos es una base de datos que contiene la información que, normalmente, representa el historial empresarial de una organización. Estos datos históricos se utilizan para realizar análisis que apoyen las decisiones empresariales a diferentes niveles, desde el diseño estratégico a la evaluación del rendimiento de una unidad determinada de la organización. Los datos contenidos en un almacén de datos se encuentran organizados para

permitir el análisis más que para procesar transacciones en tiempo real como ocurre con los sistemas de transacciones en línea.

La tecnología OLAP permite un uso más eficaz de los almacenes de datos para el análisis en línea, lo que proporciona respuestas rápidas a consultas analíticas complejas e iterativas. Los modelos de datos multidimensionales de OLAP y las técnicas de agregados de datos organizan y resumen grandes cantidades de datos para que puedan ser evaluados con rapidez mediante el análisis en línea y las herramientas gráficas. La respuesta a una consulta realizada sobre datos históricos a menudo suele conducir a consultas posteriores en las que el analista busca respuestas más concretas o explora posibilidades. Los sistemas OLAP proporcionan la velocidad y la flexibilidad necesarias para dar apoyo al analista en tiempo real.

En el diagrama expuesto a continuación se ilustra la interacción entre los almacenes de datos y la tecnología OLAP:



2.2.3 Estructuras básicas de las bases de datos multidimensionales

Las estructuras básicas de OLAP son dimensiones, medidas, jerarquías, niveles, cubos y celdas.

Las medidas son los datos que se quieren analizar, y las dimensiones definen la organización de dichas medidas.

Los elementos de una dimensión son llamados miembros. Las dimensiones organizan sus miembros en jerarquías.

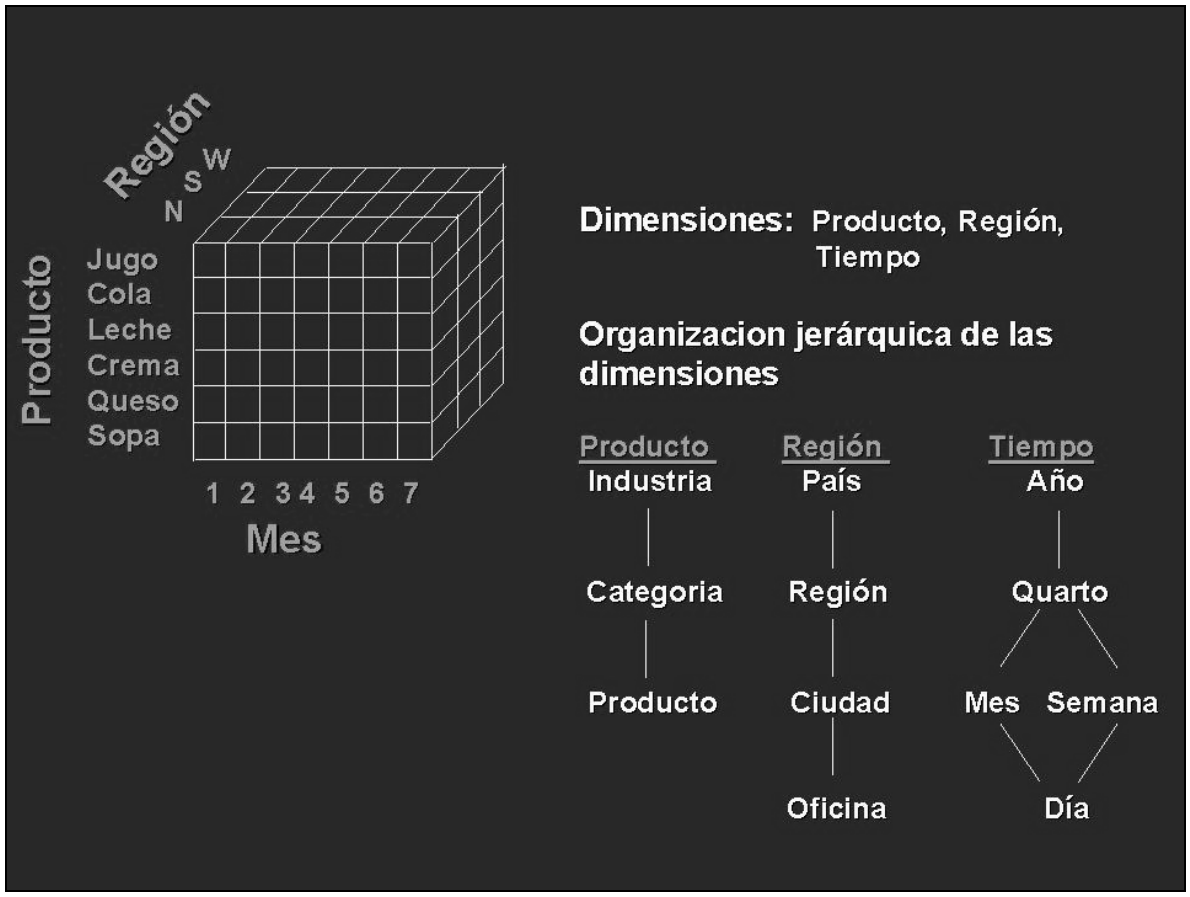
Los niveles son elementos de una jerarquía de dimensiones. Describen el orden de una dimensión desde el nivel más alto (más resumido) al más bajo (más detallado).

Cada dimensión está compuesta por uno o más niveles, cada uno de los cuales contiene uno o más miembros de la dimensión. Si una dimensión tiene más de un nivel, entonces, los miembros de la dimensión pueden ser organizados en una o más jerarquías. Una dimensión con un solo nivel no puede ser jerárquica. Cada jerarquía está completamente conectada a través de sus niveles, esto significa que cuando una dimensión tiene dos o más niveles, cada miembro del nivel más profundo (nivel "hoja") tiene un padre, cada miembro del nivel superior tiene uno o más hijos, y en cada nivel que quede en el medio (si hay alguno) cada uno de sus miembros tiene un padre y uno o más hijos.

Un cubo es un subconjunto de datos, normalmente construido a partir de un almacén de datos, organizado y resumido en una estructura multidimensional definida mediante un conjunto de dimensiones y medidas. La combinación de miembros para cada dimensión del cubo define el espacio lógico donde los valores de las medidas pueden aparecer.

Cada intersección única compuesta por un miembro de cada dimensión en el cubo es llamada celda.

En el siguiente diagrama se proporciona un ejemplo donde se puede apreciar un cubo junto con sus dimensiones y como están organizadas éstas en jerarquías:



2.2.4 Beneficios del uso de OLAP

Existen importantes diferencias de performance entre utilizar bases de datos relacionales y bases de datos multidimensionales para el análisis de datos. Manipulaciones de datos que habitualmente requieren minutos en llevarse a cabo en ambientes relacionales, en muchos casos requieren solo segundos en ambientes multidimensionales.

Las ventajas de performance ofrecidas por la tecnología multidimensional facilitan el desarrollo de herramientas de ayuda para la toma de decisiones, el cual no podría ser práctico en un ambiente relacional.

Entre las ventajas que ofrece el uso de bases de datos multidimensionales se encuentran:

- **Facilidad para la representación de los datos y exploración de los mismos**

  La salida natural de las bases de datos multidimensionales es un vista lógica de los datos. Para lograr los mismos resultados en bases de datos relacionales, el usuario debería utilizar complejas instrucciones SQL (o el uso de generadores SQL), además, en algunos casos las salidas simplemente no se podrían generar.

- **Facilidad de mantenimiento**

  Las bases de datos multidimensionales se caracterizan por su fácil mantenimiento. Esto es debido a que sus datos están almacenados en la misma forma en que se ven. Para lograr un nivel de facilidad similar, las bases de datos relacionales deben recurrir al uso de índices y de joins, elementos que dificultan el mantenimiento.

- **Performance**

  Las bases de datos multidimensionales alcanzan niveles de performance difícilmente alcanzables en un ambiente relacional.

La mayoría de las empresas tienen una limitada cantidad de tiempo y recursos para destinar al análisis de datos. En estos casos, es importante que los conjuntos de datos altamente relacionados sean ubicados en una estructura multidimensional, para facilitar el acceso y el análisis de los mismos.

Algunos ejemplos de situaciones en donde es ideal el uso de la tecnología multidimensional se listan a continuación:

- Análisis y reportes de finanzas
- Análisis de encuestas
- Control de calidad
- Análisis de las ventas de un producto

CAPÍTULO 3

Análisis

3. Análisis

3.1 Introducción

En este capítulo se expone el alcance del sistema desarrollado, quedando determinado que podrá ser capaz de hacer y que no.

Será presentada aquí una visión resumida de las cualidades del producto; proporcionando una descripción de las perspectivas, sus funcionalidades, así como las características de los potenciales usuarios.
También se incluye un resumen de las restricciones a las que está sujeto el diseño. Finalmente se mencionan los factores que pueden afectar los requerimientos a los que se hace alusión.

Las conclusiones sobre los temas mencionados fueron derivadas del estudio de las herramientas utilizadas y de las charlas con el responsable del proyecto por parte de Infocorp.

3.2 Perspectiva del producto

EL producto a desarrollar será una aplicación que funcionará sobre OLAP Services de Microsoft SQL Server, ampliando las funcionalidades que este otorga. OLAP Services es un servidor de nivel intermedio para el proceso analítico en línea, incluye un servidor que construye cubos multidimensionales de datos para su análisis y proporciona al cliente un rápido acceso a la información almacenada en el cubo. Información adicional acerca de Microsoft OLAP Services puede ser encontrada en el Apéndice A de este informe.

La nueva funcionalidad añadirá a las facultades de OLAP Services la posibilidad de exportar un cubo multidimensional de un servidor a otro, así como llevar dicho cubo a un formato que permita que sea accesible desde otros servidores multidimensionales (este nuevo formato será representado en XML).

3.3 Herramientas utilizadas

Desde un comienzo, el proyecto tenía pautado que su desarrollo se llevara a cabo con algún lenguaje Microsoft Visual, y Microsoft Visual Basic 6.0 fue el seleccionado.

Otro elemento preestablecido era la utilización de XML como leguaje a utilizar para la representación de la metadata de los cubos luego de exportada. Esta elección se debe en parte a la importancia que está empezando a tener el leguaje, sus perspectivas a futuro, su capacidad expresiva, además del hecho de la necesidad de proporcionar un formato abierto para la estructura exportada (elemento especialmente importante en este proyecto).

Para la comunicación con Microsoft OLAP Services fue utilizado DSO *(Decision Support Objects)*, que es un modelo de objetos proporcionado por OLAP Services que cuenta con un conjunto de interfaces, objetos, colecciones, métodos y propiedades, que permiten la manipulación de las estructuras multidimensionales presentes en el servidor OLAP.

Para la tarea de migración de bases de datos entre distintos servidores, se hizo uso de los paquetes de Servicios de Transformación de Datos (DTS) que incluye Microsoft SQL Server 7.0. Cada paquete DTS define una o más tareas que se ejecutan en una secuencia coordinada, proporcionando funciones de migración y transformación de datos.

La interacción con XML se realizó a través del Modelo de Objetos de Documentos XML (DOM). Mediante este modelo de objetos es posible cargar archivos XML, recolectar información de éstos archivos, así como explorarlos y manipularlos. DOM cuenta con objetos que exponen métodos y propiedades que le posibilitan al programador trabajar con la información presente en una instancia XML.

3.4 Funcionalidades

La migración de un cubo de un servidor a otro se realizará en dos etapas: primero el cubo es llevado a un formato intermedio (cubo exportado), y luego se lleva de ese estado intermedio al servidor destino, reconstruyendo el cubo al formato utilizado por OLAP Services.

A continuación se detallan las funcionalidades proporcionadas por el producto desarrollado:

- **Exportar un cubo regular**

  Esta operación le permite al usuario exportar un cubo regular que reside en un determinado servidor al nuevo formato especificado en XML. Teniendo la libertad de especificar también el lugar de residencia del cubo exportado.

- **Exportar un cubo virtual**

  El usuario aquí tiene básicamente las mismas posibilidades que en la operación anterior (pero en este caso el cubo exportado es virtual). Además, en esta operación se puede manifestar si se desea o no exportar los cubos base asociados al cubo virtual a exportar.

- **Importar cubo regular**

  Esta operación consiste en importar un cubo regular expresado en el formato XML a un servidor especificado, obteniendo un cubo en la representación utilizada por Microsoft OLAP Services. En el momento de realizar la importación se puede expresar si se quiere importar la base de datos en la que está basado el cubo en cuestión o si solo se importará su metadata.

- **Importar cubo virtual**

  Aquí se realiza la importación de un cubo virtual, previamente exportado, a un servidor OLAP Services. El usuario tiene la posibilidad de realizar también la importación de los cubos base correspondientes, siempre que estos se hayan incluido en el proceso de exportación. Para los distintos cubos base, se podrá seleccionar si es deseado importar sus bases de datos.

- **Generar un archivo para ser visualizado en un browser**

  Esta operación lleva la metadata de un cubo a un archivo HTML, de forma de que ésta pueda ser visualizada luego en un browser estándar. La realización de esta operación requiere que el cubo se haya exportado previamente.

Las operaciones que se listan a continuación permiten realizar cambios en la metadata de un cubo, estos cambios se realizan a nivel de la interfaz de la aplicación y pueden o no ser reflejados luego en la representación en formato XML del cubo correspondiente.

- **Eliminar un elemento de un cubo**

  Esta operación permite eliminar elementos de la metadata de un cubo. La diversidad de elementos con posibilidad de eliminar depende del hecho de si se está trabajando sobre un cubo regular o virtual.

- **Renombrar un elemento de un cubo**

  Haciendo uso de esta operación se pueden renombrar los elementos de la metadata de un cubo. La diversidad de elementos con posibilidad de renombrar depende del hecho de si se está trabajando sobre un cubo regular o virtual.

- **Cambiar la descripción de un elemento de un cubo**

  Mediante esta operación es posible cambiar la descripción de los elementos de la metadata de un cubo o agregar una a los que no la posean. La diversidad de elementos con posibilidad de modificar su descripción depende del hecho de si se está trabajando sobre un cubo regular o virtual.

- **Especificar si una dimensión es compartida o privada**

  Dada una dimensión de un cubo, esta operación permite transformarla a privada, en caso de que sea compartida, o llevarla a compartida en caso de que sea privada. Esta operación solo se podrá llevar a cabo si se está trabajando sobre un cubo regular.

- **Especificar si una medida es oculta o no**

  Dada una medida de un cubo, haciendo uso de esta operación se podrá especificar si se desea que la medida sea oculta o no. Esta operación solo se podrá llevar a cabo si se está trabajando sobre un cubo regular.

- **Especificar el modo de almacenamiento de una partición**

  Con esta operación es posible indicar el modo de almacenamiento deseado para una partición, pudiendo ser éste MOLAP, HOLAP o ROLAP.

3.5 Opciones descartadas

En el transcurso de la etapa de análisis algunas funcionalidades deseables fueron dejadas de lado. A continuación se listan las que consideramos de mayor importancia:

- **No se permite modificar la mayoría de los elementos de la metadata de un cubo virtual**

  Esta decisión se tomó porque tanto las dimensiones como las medidas utilizadas por un cubo virtual, son elementos de la metadata de alguno de sus cubos base. Por lo tanto, no nos pareció conveniente que desde el cubo virtual se tenga la potestad para realizar modificaciones a estos elementos (ya que esto implicaría cambios en sus cubos base, o dejar a estos inválidos).

- **No se permite modificar la mayoría de las propiedades de los elementos que componen la estructura del cubo**

  De la amplia galería de propiedades que exhiben los objetos de las estructuras multidimensionales manejadas por OLAP Services, solo se permite modificar un subconjunto de éstas. La razón de este impedimento se debe a las restricciones en cuanto a tiempo de desarrollo a las que nos vimos envueltos, y al hecho de que estas modificaciones se pueden hacer en Microsoft OLAP Services, por lo que no consideramos como un hecho de mayor relevancia su inclusión.

- **Las únicas bases de datos que se pueden migrar son las que se encuentran en servidores SQL Server**

  Como se mencionó anteriormente en este capítulo, para la migración de las bases de datos en las que están basados los cubos se hizo uso de DTS. Utilizar DTS implica una implementación que varía dependiendo del proveedor origen y destino de la base de datos. Teniendo en cuenta además la restricción dada por la fecha de entrega del proyecto, y el hecho de que en la empresa Infocorp se nos informó que los cubos con los que ellos trabajan se encuentran en bases de datos que están en servidores SQL Server, llegamos a tomar la decisión de implementar una solución que solo sea capaz de migrar bases de datos entre servidores SQL Server. Esto implica que si un cubo está basado en una base de datos que no se encuentra en un servidor SQL Server, ésta no podrá ser migrada (pero sí podrá ser migrado el cubo correspondiente).

- **Independientemente del modo de almacenamiento de las particiones en que se basa el cubo, se requiere que para su procesamiento estén accesibles las bases de datos en que se basan sus particiones**

  En una primera instancia parecía particularmente deseable la idea de que, en los casos de cubos cuyas particiones utilizaran un modo de almacenamiento MOLAP, no fuera necesario que desde el servidor destino de la importación del cubo estuviera accesible la base de datos en que se basa dicho cubo para permitir su procesamiento. Este hecho se basaba en la idea de reconstruir directamente la estructura multidimensional en la que están organizados los datos del cubo en el servidor OLAP destino. La realización de esta tarea implica un conocimiento detallado acerca de cómo están organizados los datos en OLAP Services, y la información acerca de esta organización no está a nuestra disposición. Además, si la tuviéramos, esta tarea implicaría la utilización de importantes recursos en cuanto a tiempo de desarrollo. Todos estos elementos nos llevaron a tomar la decisión de imponer que siempre sea necesario que la base de datos en que se basa el cubo esté accesible desde el servidor donde este se encuentre, para poder realizar el procesamiento del cubo.

- **No es posible agregar elementos al cubo**

  La no inclusión de una operación que permita agregar elementos a un cubo se debe básicamente a las limitaciones en cuanto a plazos establecidos. Puesto que estas tareas son admitidas desde OLAP Services, el hecho de duplicar estas funcionalidades no fue considerado de mayor interés.

- **Si un cubo virtual se importa con sus cubos base, éstos no pueden sufrir modificaciones**

  Esta decisión se fundamenta en que modificaciones a los cubos base pueden llevar a dejar inválido el cubo virtual que se vincula a ellos, y dado que éstos cubos regulares se exportaron en una operación de exportación de su cubo virtual, no creímos razonable permitir realizar operaciones que puedan dejar inválido dicho cubo.

3.6 Características del usuario

El producto desarrollado no está destinado para que sea utilizado por todo tipo de personas.

Los eventuales usuarios del producto deberán poseer conocimientos de estructuras multidimensionales, conocedores de la forma que OLAP Services implementa este tipo de estructuras, modos de almacenamiento, diseño de agregaciones, etc. Además es recomendable que conozca la información representada en el cubo que se está exportando, ya que las modificaciones que puedan llevarse a cabo a dicho cubo pueden tener una repercusión importante en los análisis para los que está destinado a facilitar el cubo en cuestión.

3.7 Restricciones generales

Como se mencionó en puntos anteriores, el producto funcionará sobre OLAP Services, utilizando la interfaz que este provee para la manipulación de sus estructuras (DSO). Por lo tanto, la aplicación desarrollada está restringida por la potencia de estas interfaces.

Algo similar a lo señalado con DSO ocurre con el modelo de objetos utilizado para la interacción con XML (DOM) y con los paquetes de transformación de datos (DTS) de Microsoft SQL Server.

Un ejemplo de estas restricciones son las que aparecen con DOM en el caso de los tipos de datos; estos no siempre son los utilizados por Visual Basic o el propio DSO. Dándose el caso de que en algunas ocasiones tipos de datos para representar números, como Integer o Long, tengan rangos que varíen respecto a Visual Basic y DSO.

Otras restricciones aparecen por la utilización de XML como lenguaje de representación de las estructuras exportadas. Con este lenguaje se hace muy difícil imponer las restricciones que deberían estar presentes entre los objetos que componen un cubo (restricciones que sí están expresadas en DSO). Ejemplos de estas restricciones son: la unicidad de nombres de dimensiones y medidas (entre otros objetos), la vinculación entre el nombre de un cubo y el nombre de su partición primaria, la validez de los valores de las propiedades de los distintos elementos, expresiones MDX, etc.

También cabe destacar que el orden en que aparecen los elementos de las estructuras multidimensionales en un archivo XML es un orden fijo preestablecido. No permitiendo que los elementos puedan aparecer en otro orden que el determinado a través del esquema utilizado por la instancia.

3.8 Asunciones y dependencias

No se ha alcanzado un estándar respecto a un modelo para la representación de estructuras multidimensionales. Existen diversas herramientas OLAP con elementos en común y otros en los que difieren unas de otras.

La aplicación desarrollada supone que se está trabajando con la estructura utilizada por Microsoft OLAP Services. Si en futuras versiones de OLAP Services la representación de cubos cambia, la aplicación no será capaz de funcionar correctamente.

CAPÍTULO 4

Diseño

4. Diseño

4.1 Introducción

En el presente capítulo, se pretende describir la estructura elemental del producto. Se expondrá una visión en conjunto de todos sus componentes e interacción, de forma de proporcionar al lector un entendimiento completo de los aspectos relevantes al diseño del sistema. La información aquí contenida puede ser complementada con los Apéndices, para lograr un conocimiento más profundo de los términos a los que se hace referencia.

Se intentará seguir una metodología top-down en la presentación, de esta manera resulta relativamente sencillo comprender el análisis efectuado así como las motivaciones que llevaron al diseño actual del prototipo.

4.2 Panorama global de diseño

Como se mencionó en capítulos previos, el sistema desarrollado interactúa con distintas tecnologías para llevar a cabo los objetivos planteados.
El diseño, desde un comienzo, se orientó a enmarcar estas tecnologías disímiles, en entidades o módulos independientes. De esta manera, es posible contar con un conjunto de primitivas más sencillo y adecuado a nuestros propósitos, facilitando enormemente las tareas críticas de la aplicación.

Esta estrategia, no solo provoca que el diseño sea lo suficientemente flexible como para adaptar eventuales cambios con la mínima dificultad, sino que también favorece el proceso de implementación y permite incorporar nuevas funcionalidades al producto, sin afectar el comportamiento general del mismo.

En consecuencia, el modelo de diseño seleccionado se basa en la metodología de orientación a objetos y en la definición de unidades lógicas o componentes aplicables a los distintos contextos de interacción.

Conceptualmente, la solución propuesta se compone de cuatro entidades u objetos principales:

- Una entidad especializada en el intercambio de información con Microsft OLAP Services, más específicamente, esta entidad proporciona un conjunto de métodos de

mediana complejidad que interactúan directamente con la interfaz de programación que OLAP Services brinda a las aplicaciones.

- Una segunda entidad responsable del acceso al almacén de datos vinculado a cada cubo.

- Una estructura multidimensional local, conocida únicamente por la aplicación, sobre la que se efectúan todas las tareas de edición que el usuario determine.

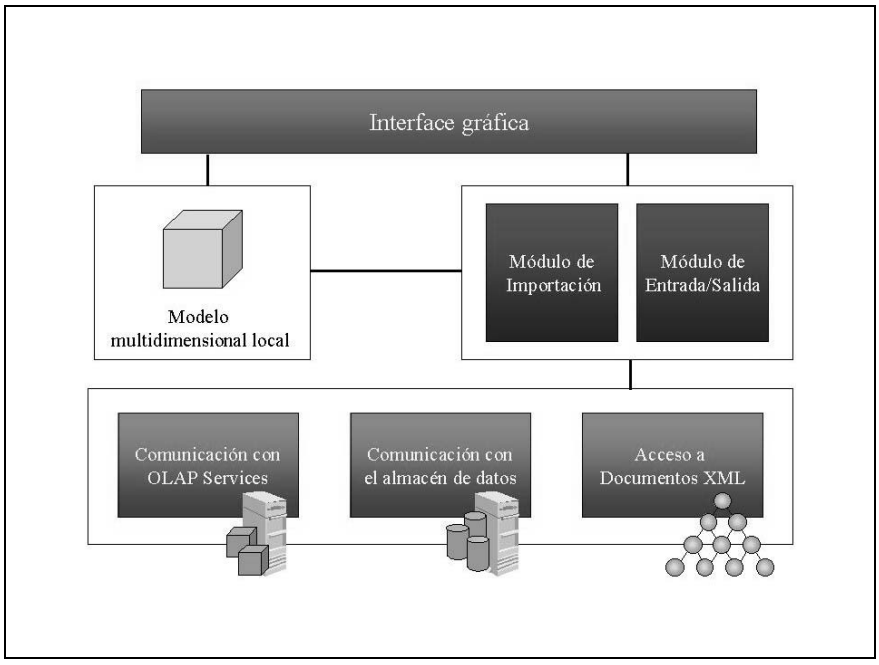
- Y finalmente, una entidad encargada de la generación, carga y transformación de documentos con formato XML.

El proceso de migración de datos y metadatos, es llevado a cabo por otros dos componentes que utilizan los servicios previos para lograr su cometido. Estos son:

- Módulo de importación, que permite la reconstrucción total del cubo (esto es, metadata y datos) en el Servidor OLAP destino.

- Módulo de entrada/salida (IO), que proporciona acceso al sistema de archivos local, facilitando la generación y carga de documentos XML.

Las entidades y módulos descriptos, en conjunto con una interface gráfica independiente (MMI), conforman el sistema implementado.



4.3 Arquitectura general y principales entidades

Esta sección introduce a los conceptos elementales sobre las principales tecnologías con las cuales se comunica la aplicación, presentando las entidades u objetos que las accesan.

La intención de este apartado, no es profundizar sobre cada componente, sino exponer una muy breve reseña de las ideas fundamentales y motivaciones que llevaron a su consideración, en un intento de familiarizar al lector con los conceptos clave manejados.

Para tal fin, se presentarán las nociones que definen el producto evitando entrar en los detalles que esconden. Por consiguiente, puede que algunos conceptos permanezcan en sombra por no haber profundizado en ellos. En cualquier caso, los capítulos posteriores despejarán estas y otras incógnitas, ya que su principal cometido es efectuar un análisis detallado del proceso de implementación.

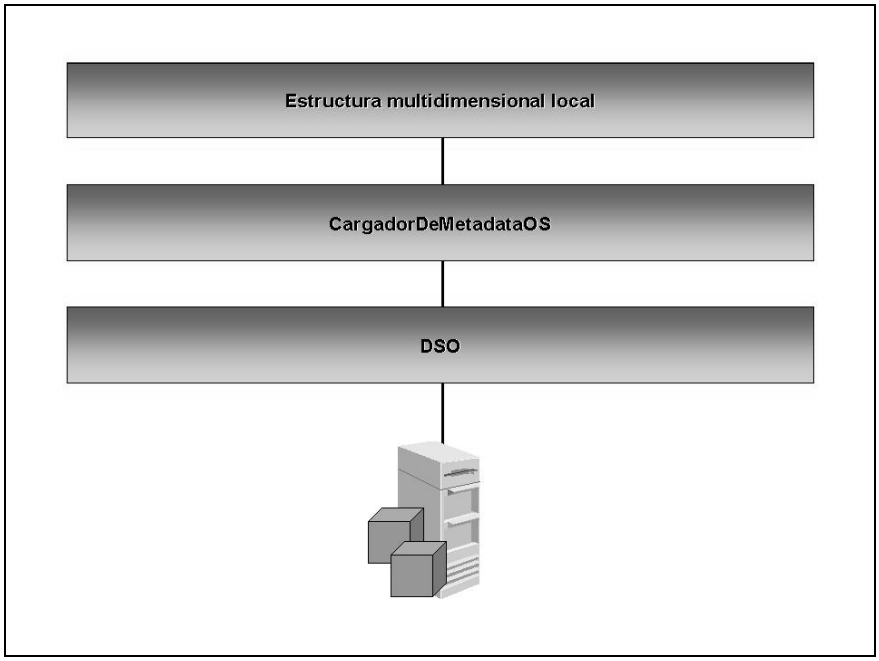
4.3.1 Acceso a la estructura multidimensional de OLAP Services

Los Servicios OLAP de Microsoft SQL Server ofrecen herramientas que le permiten extender su funcionalidad mediante programación. El modelo de objetos de servidor, denominado Objetos de ayuda para la toma de decisiones (DSO, *Decision Support Objects*), puede utilizarse para crear aplicaciones que definan y administren cubos y otros objetos.
Esta interface de programación se proporciona esencialmente para permitir que las aplicaciones personalizadas interactúen con el modelo de objetos que controla el servidor, en otras palabras, DSO expone el modelo de objetos del Servidor OLAP.

El acceso a este modelo por parte de nuestra aplicación, está contenido en un único objeto, dedicado exclusivamente a proporcionar las rutinas y servicios necesarios para tal fin. Las ventajas de esta metodología de diseño son muchas, seguramente la de mayor peso sea una distribución más eficiente de la complejidad, orientando el sistema hacia una composición modular bien definida y poco acoplada. Otra característica destacada es el poder de abstracción que se brinda y la capacidad de evolución del producto desarrollado, al cambiar las estructuras jerárquicas que implementa Microsoft en su modelo.

Básicamente este objeto, denominado *CargadorDeMetadataOS*, consiste en una clase que provee facilidades para establecer y liberar una conexión con el Servidor OLAP que se especifique y trabajar directamente con el modelo de objetos que presenta DSO. De esta forma, es posible reconstruir el esquema multidimensional que utiliza Microsoft a partir del nuestro y realizar el proceso inverso, es decir, cargar nuestro propio modelo de objetos con la información contenida en la estructura jerárquica que definen los Servicios OLAP.

Al limitar estos paradigmas de interacción a una única entidad, se logra independizar la implementación obteniendo un mayor nivel de reutilización y flexibilidad en el diseño.



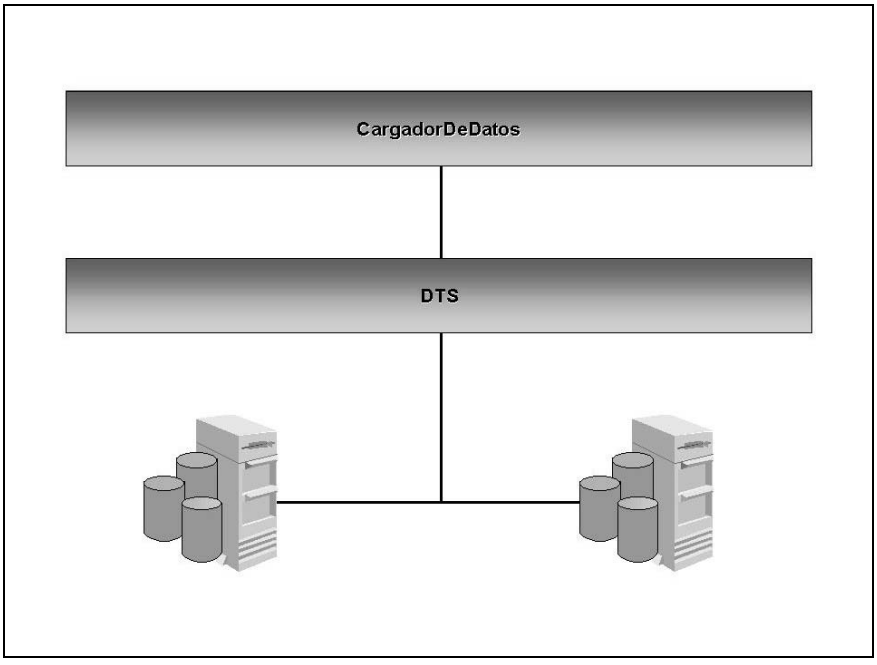
4.3.2 Acceso al almacén de datos

Los Servicios de transformación de datos (DTS, *Data Transformation Services*) permiten importar, exportar o transformar datos en un único proceso. La definición de este proceso puede guardarse en un objeto que se denomina *paquete*. Cada paquete define un flujo de trabajo que incluye una o más tareas que se ejecutan en una secuencia coordinada de pasos. Las tareas pueden copiar datos de un origen a un destino, transformar datos mediante una secuencia de comandos, ejecutar una instrucción SQL en el servidor o incluso ejecutar un programa externo. Una tarea de transferencia de objetos de SQL Server también puede transferir objetos de bases de datos entre equipos en los que se ejecuta Microsoft SQL Server versión 7.0.

Entre los componentes de SQL Server que se utilizan para definir paquetes, se incluyen las interfaces de programación DTS. Los DTS proporcionan un conjunto de interfaces de Automatización OLE y un conjunto de interfaces COM (*Component Object Model*) que se pueden utilizar para crear aplicaciones personalizadas para importación, exportación y transformación en sistemas de desarrollo que admitan Automatización OLE o COM (como es el caso de Microsoft Visual Basic).

Para construir un paquete, es necesario configurar las conexiones de los orígenes y destinos de los datos, definir las asignaciones de transformación, configurar las propiedades de las tareas y establecer las relaciones de precedencia.

Este proceso en nuestra aplicación, es consolidado en un objeto denominado *CargadorDeDatos*, que utiliza los servicios proporcionados por los DTS y permite migrar en forma trasparente, el almacén de datos de un cubo, del servidor origen al destino requerido.

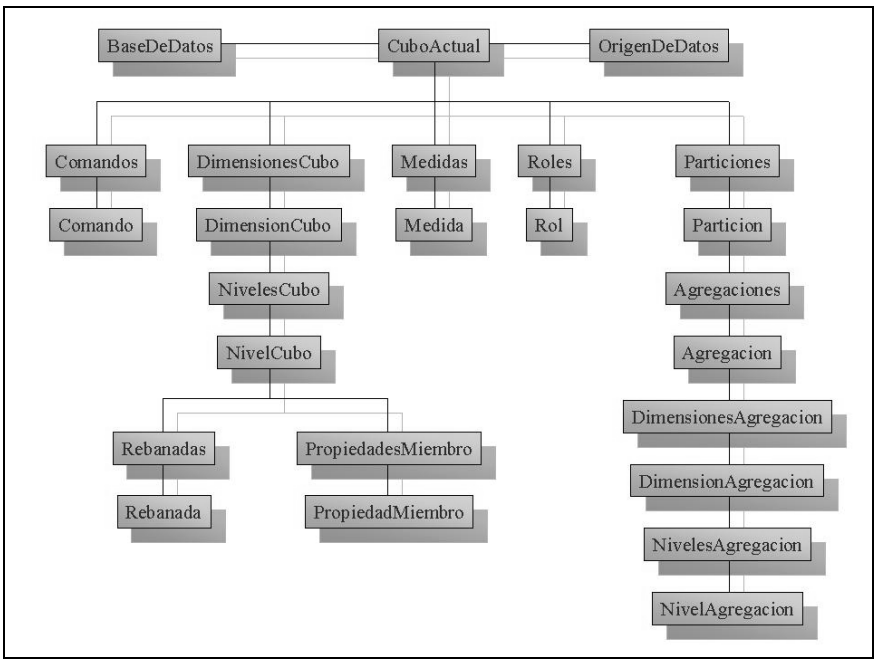


4.3.3 Modelo multidimensional local

El modelo desarrollado, surge de la conexión entre diversos objetos sencillos, colecciones, métodos y propiedades. Grupos de estos objetos de menor complejidad, son dispuestos jerárquicamente para definir los elementos primordiales de la estructura multidimensional, que tiene al objeto *CuboActual* como raíz.

Este ordenamiento jerárquico se compone de dos objetos y cinco colecciones que a su vez definen su propia agrupación subordinada.

A continuación se muestra una vista expandida de esta estructura, incluyendo los miembros básicos de las colecciones que son clave para el almacenamiento y definición de la metadata del cubo sobre el que se trabaja (de allí la denominación “CuboActual”).



Como es de suponer, los objetivos básicos de esta estructura son tres:

- Debe almacenar toda la información requerida para su posterior reconstrucción en el Servidor OLAP.
- Debe facilitar su representación en documentos con formato XML.
- Debe permitir a nuestra aplicación efectuar todas las tareas de edición que el usuario especifique, manteniendo la consistencia necesaria para cumplir el primer objetivo.

De esta manera se logra la autonomía buscada, puesto que solo será preciso escribir las rutinas apropiadas que permitan la inicialización de esta estructura y finalmente las que permitan reconstruir este modelo propio en la estructura multidimensional propuesta por Microsoft.

4.3.4 Representación en XML

XML, lenguaje extensible de etiquetas (extensible por que no es un formato prefijado como por ejemplo HTML), describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas que los procesan.

XML es un *perfil de aplicación* o una forma restringida de SGML, el Lenguaje Estándar Generalizado de Marcación. Por construcción, los documentos XML son documentos SGML conformados.

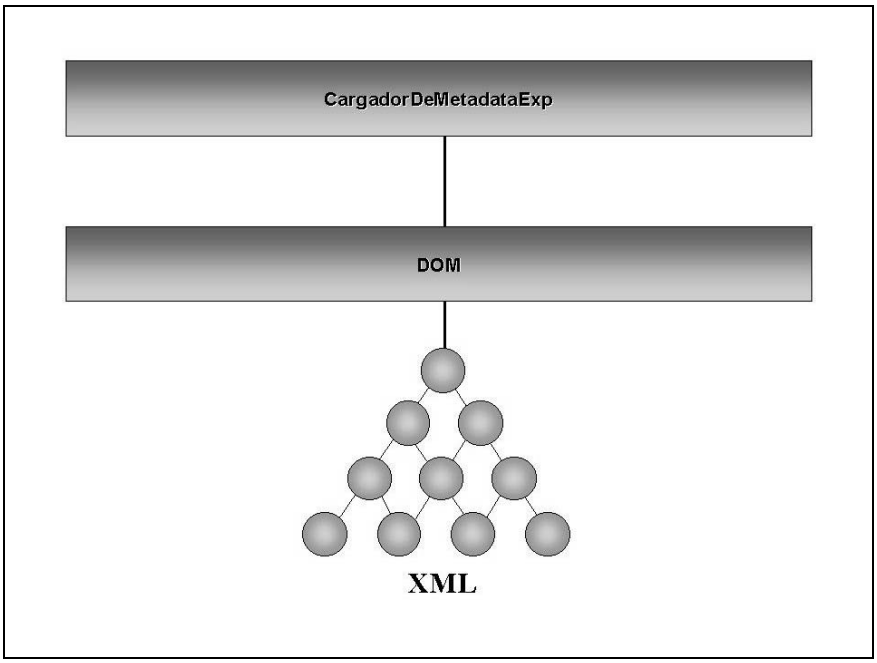
XML por tanto es ante todo un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos. Los documentos XML se componen de unidades de almacenamiento llamadas entidades, que contienen datos analizados o sin analizar. Los datos analizados se componen de caracteres, algunos de los cuales forman los datos del documento y el resto forman las etiquetas. Estas etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento.

XML proporciona un mecanismo para imponer restricciones en esta estructura lógica y de almacenamiento. Cualquier aplicación que trabaje con XML necesita un módulo software llamado procesador *(parser)* XML. Su función es leer documentos y proporcionar acceso a su contenido y estructura. Para poder llevar a cabo esta función, la aplicación debe proporcionar información al procesador XML de como se encuentra almacenado el documento a través de, por ejemplo, un esquema *(schema)*. El esquema proporciona la gramática para una clase de documentos XML. Esta gramática contiene la definición del conjunto de etiquetas que puede contener esa clase de documentos XML. Un esquema es generalmente un archivo (o varios usados conjuntamente) que contiene la definición formal de un tipo de documento particular. Este define los nombres que pueden utilizarse en los elementos, donde pueden aparecer y como se interrelacionan. Se profundizará sobre estos y otros temas referentes a XML en posteriores capítulos.

Con el modelo de objetos de documento XML (DOM, *Document Object Model*), es posible cargar, procesar, navegar y manipular archivos XML. El modelo de objetos DOM, expone el documento XML como una estructura jerárquica compuesta por múltiples nodos; la interface de programación que se proporciona permite a las aplicaciones recorrer esta estructura de árbol y manipular sus nodos.

Nuevamente, para ganar en simplicidad, la interacción entre DOM y nuestra aplicación se reduce a un solo objeto, denominado *CargadorDeMetadataExp*.

Se podría decir, que esta entidad cumple un propósito similar al *CargadorDeMetadataOS* ya descripto, puesto que su labor principal consiste en convertir nuestra estructura multidimensional local *(CuboActual)* en una instancia XML válida y efectuar el proceso inverso, es decir, reconstruir esta estructura a partir de una instancia XML dada.

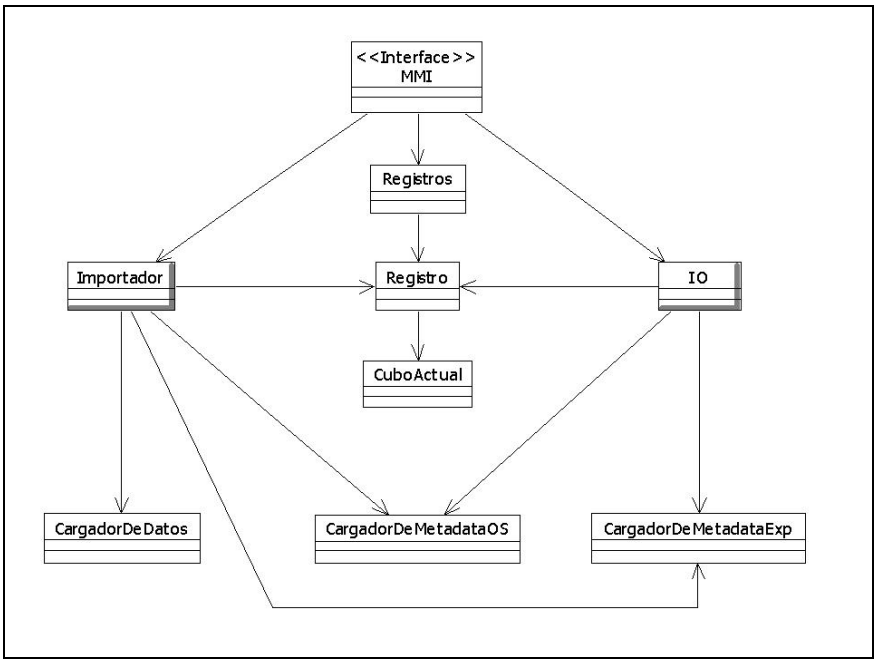


Otro aspecto relevante de este componente, es que proporciona los medios para transformar una instancia XML, que represente la metadata de un cubo, en un documento HTM.

La transformación se logra aplicando hojas de estilo *(XSL, Extensible Stylesheet Language)* previamente desarrolladas, con el propósito de formatear los datos del documento XML en una representación más amigable en HTM. Esta representación, posteriormente podrá ser utilizada para visualizar la información del cubo en un browser estándar.

4.4 Descripción de dependencias

El modelo de descripción del diseño utilizado, puede entonces ser expresado como una colección de componentes, cada uno de las cuales contendrá las relaciones que se señalan a continuación.



Aquí se puede observar, como la interface gráfica *(MMI)* administra múltiples cubos a través de una colección de registros. Esta clase Registro, almacena la información que identifica al origen del cubo (es decir, si proviene de un Servidor OLAP o de un documento XML). Esto es imprescindible a la hora de discriminar el conjunto de operaciones que se puede aplicar al cubo sobre el cual la interface trabaja.

4.5 Descripción de interfaces

La notación que introducimos aquí describe la arquitectura del producto, especificando las funcionalidades o servicios que cada objeto y módulo del mismo implementa.

Se utiliza una variación de TDN *(Textual Design Notation)*, para suministrar una descripción abstracta que facilite a cualquier programador comprender la estructura detallada del sistema.

class CargadorDeMetadataOS
uses DSO, CuboActual;

exports

procedure conectar(strNomServidor: **in** String);
Se conecta con el servidor cuyo nombre es pasado como parámetro.

function conectado(): Boolean;
Devuelve True si existe una conexión con un servidor y False en caso contrario.

procedure desconectar();
Se desconecta del servidor.

function getNomServidor(): String;
Devuelve el nombre del servidor.

function listarCubos(): Coleccion;
Lista los cubos del servidor.
Esto lo hace devolviendo una colección de colecciones, conteniendo estas la lista de cubos asociado a cada base de datos del servidor.

procedure verificoCuboVirtual(virtual: **in** CuboActual,
strNomBDatos: **in** String);
Verifica que un cubo virtual pueda ser construido en el servidor al cual existe conexión.

function buscarCubo(strNomCubo: **in** String,
strNomBDatos: **in** String): DSO.MDStore;
Devuelve el cubo cuyo nombre es pasado como parámetro, y que corresponde a la base de datos de nombre el pasado en la variable strNomBDatos.

function buscarBDatos(strNomBDatos: **in** String): DSO.MDStore;
Devuelve la base de datos cuyo nombre es pasado como parámetro.

function generarCuboActual(strNomCubo: **in** String,
strNomBDatos: **in** String): CuboActual;
Genera un objeto de tipo CuboActual, a partir de un cubo de OLAP Services cuyo nombre es pasado como parámetro.

function existeBDatos(strNomBDatos: **in** String): Boolean;
Devuelvo True si la base de datos se encuentra en el servidor y False en caso contrario.

procedure generarBaseDeDatos(bDatos: **in** CuboActual.BaseDeDatos);
Genera la base de datos pasada como parámetro en un servidor OLAP Services.

Function existeOrigenDeDatos(nomBDatos: **in** String,
nomOrigen **in** String): Boolean;
Retorna True si el origen de datos de nombre nomOrigen se encuentra en la base de datos de nombre nomBDatos y False en caso contrario.

procedure generarOrigenDeDatos(origen: **in** CuboActual.OrigenDeDatos,
strNomBDatos: **in** String);
Genera el origen de datos pasado como parámetro en un servidor OLAP Services. Ubicándolo en la base de datos cuyo nombre es pasado como parámetro.
En caso de que la base de datos no exista, o que exista pero ya tiene un origen de datos con el mismo nombre que el pasado como parámetro, el nuevo origen no es creado.

procedure generarBDyDS(unCuboActual: **in** CuboActual);
Genera la base de datos y el origen de datos especificados en el objeto de tipo CuboActual pasado como parámetro.

procedure generarCuboOS(unCuboActual: **in** CuboActual);
Genera un cubo de OLAP Services a partir de un objeto de tipo CuboActual pasado como parámetro.

attributes *Atributos relevantes.*

Servidor: DSO.Server;
Servidor al cual se mantiene una conexión.
end CargadorDeMetadataOS

class CargadorDeDatos
uses DTS.Package;

exports

procedure crearPaquete(strNomServidor: **in** String, strNomBDatos: **in** String, strUsuario: **in** String, strContraseña: **in** String);
Crea un paquete DTS.

procedure ejecutarPaquete(strNomServidor: **in** String, strNomBDatos: **in** String, strUsuario: **in** String, strContraseña: **in** String);
Ejecuta el paquete DTS paqueteDatos.

function listarBDatos(strNomServidor: **in** String, strUsuario: **in** String, strContraseña: **in** String): Coleccion;
Lista las bases de datos registradas en el servidor cuyo nombre es pasado como parámetro.

procedure verificarConexion(strNomServidor: **in** String, strUsuario: **in** String, strContraseña: **in** String);
Verifica que sea posible establecer una conexión con el servidor cuyos datos son pasados como parámetros.

attributes *Atributos relevantes.*

paqueteDatos: DTS.Package;
Paquete DTS
end CargadorDeDatos

class CargadorDeMetadataExp
uses DOM, CuboActual;

exports

function generarCuboActual(strNomArchOrigen: **in** String): CuboActual;
Devuelve una referencia a un objeto de tipo CuboActual, creado a partir de la información contenida en la instancia XML almacenada bajo el nombre strNomArchOrigen.

procedure generarCuboExp(strNomArchDestino: **in** String, cubo: **in** CuboActual)
Crea una instancia XML con la información de cubo y la almacena bajo el nombre indicado en strNomArchDestino.

procedure transformar(strRutaXML: **in** String, strRutaXSL: **in** String,
strRutaDestino: **in** String);
Transforma el documento XML (cuya ruta se indica en strRutaXML), aplicando la hoja de estilo señalada por strRutaXSL.
El resultado de la transformación se almacena bajo la ruta strRutaDestino.

attributes *Atributos relevantes.*

mstrNomArchEsquema: String;
Nombre del archivo que contiene el esquema XML.
end CargadorDeMetadataExp

module Importador
uses Registro, CargadorDeMetadataOS, CargadorDeMetadataExp, CargadorDeDatos;

exports
function importar(registroDeCubo: **in** Registro): Registro;
Permite importar en forma interactiva, el contenido de registroDeCubo al destino que el usuario especifique.
Si la importación es exitosa, se devuelve un nuevo registro con toda la información, en cualquier otro caso se retorna Nothing.
Si el procesamiento de registroDeCubo.cubo en el Servidor destino fracasa, este método considera como exitosa la importación efectuada.
end Importador

module IO
uses Registro, CuboActual, CargadorDeMetadataOS, CargadorDeMetadataExp;

exports
function abrir(optional strRutaArch: **in** String): Registro;
Abre el archivo strRutaArch. Si el parámetro no es especificado, este método permite seleccionar un archivo con formato .cub para su apertura.
Si la operación es exitosa se devuelve un registro con toda la información, en cualquier otro caso se retorna Nothing.

function guardar(strRutaBin: **in** String, cubo: **in** CuboActual): Boolean;
Permite guardar cubo, actualizando el contenido del archivo (.cub) strRutaBin.
Si el proceso es exitoso se devuelve True.

function guardarComo(registroDeCubo: **in** Registro): Registro;
Permite guardar la información contenida en registroDeCubo, bajo la nueva ubicación especificada por el usuario.
Si la operación es exitosa se devuelve un registro con toda la información, en cualquier otro caso se retorna Nothing.

function generarArchVisualizacion(registroDeCubo: **in** Registro): String;
Permite generar un Documento HTM bajo la ubicación especificada por el usuario. Este documento HTM, permitirá visualizar la estructura de registroDeCubo.cubo en un browser.
Si el proceso es exitoso, se devuelve la ruta absoluta al archivo con formato HTM generado, en cualquier otro caso se retorna una cadena de longitud cero ("").

end IO

class Registro
uses CuboActual;

exports

function getCubo(): CuboActual;
Devuelve el cubo vinculado al registro.

procedure setCubo(nuevoCubo: **in** CuboActual);
Cambia el cubo vinculado al registro, por nuevoCubo.

function getRutasCubosBase(): ColString;
Devuelve una colección, con las rutas absolutas a los archivos con formato XML, en donde están almacenados los cubos base vinculados al registro.

function getTipoOrigen(): Enumerado;
Indica el tipo de registro.

procedure setTipoOrigen(enumTipoOrigen: **in** Enumerado);
Cambia el tipo de origen del registro, por enumTipoOrigen.

function getOrigen(): String;
El origen de un registro puede ser un nombre de Servidor o una ruta absoluta a un archivo binario con extensión .cub.

procedure setOrigen(strOrigen: **in** String);
Cambia el origen del registro por strOrigen.

attributes *Atributos relevantes.*

mCubo: CuboActual;
Referencias a objetos.

mRutasCubosBase: ColString;
Colecciones.

menumTipoOrigen: Enumerado;
mstrOrigen: String;
Otras propiedades.
end Registro

class CuboActual
uses OrigenDeDatos, BaseDeDatos, Comandos, DimensionesCubo, Medidas, Roles, Particiones, ColString;

exports

function getClave(): String;
Devuelve la clave asociada al objeto.

procedure setClave(strClave: **in** String);
Cambia la clave asociada al objeto por strClave.

function getOrigenDeDatos(): OrigenDeDatos;
Devuelve el origen de datos asociado al cubo actual.

function getBaseDeDatos(): BaseDeDatos;
Devuelve la base de datos asociada al cubo actual.

function getComandos(): Comandos;
Devuelve los comandos asociados al cubo actual.

function getDimensionesCubo(): DimensionesCubo;
Devuelve las dimensiones asociadas al cubo actual.

function getMedidas(): Medidas;
Devuelve las medidas asociadas al cubo actual.

function getRoles(): Roles;
Devuelve los roles asociados al cubo actual.

function getParticiones(): Particiones;
Devuelve las particiones asociadas al cubo actual.

function getNombresCubosBase(): ColString;
Si el cubo actual es virtual, devuelve los nombres de los cubos base asociados al mismo.

function getNombre(): String;
Devuelve el nombre asociado al cubo actual.

procedure setNombre(strNombre: **in** String);
Cambia el nombre asociado al cubo actual por strNombre.
Esta operación renombrará la partición principal si esta existe (asumiendo que es el primer elemento de la colección Particiones).
Se asume, además, que strNombre es un nombre válido para este tipo de objeto.

function getDescripcion(): String;
Devuelve la descripción asociada al cubo actual.

procedure setDescripcion(strDescripcion: **in** String);
Cambia la descripción asociada al cubo actual por strDescripcion.

function getTablaDeHechos(): String;
Devuelve el nombre de la tabla de hechos asociada al cubo actual.

procedure setTablaDeHechos(strTablaDeHechos: **in** String);
Cambia el nombre de la tabla de hechos, asociada al cubo actual, por strTablaDeHechos.

function getFilasEstimadas(): Integer;
Devuelve el tamaño estimado de la tabla de hechos asociada al cubo actual.

procedure setFilasEstimadas(intFilasEstimadas: **in** Integer);
Cambia el tamaño estimado de la tabla de hechos, asociada al cubo actual, por intFilasEstimadas.

function getEsRegular(): Boolean;
Indica si el cubo actual es regular (True) o virtual (False).

procedure setEsRegular(blnEsRegular: **in** Boolean);
Cambia la propiedad esRegular, asociada al cubo actual, por blnEsRegular.

function getClausulaJoin(): String;
Devuelve la condición de join de la tabla de hechos con las tablas de dimensiones asociadas al cubo actual.

procedure setClausulaJoin(strClausulaJoin: **in** String);
Cambia la condición de join, de la tabla de hechos con las tablas de dimensiones asociadas al cubo actual, por strClausulaJoin.

function getPrefijoDeAgregacion(): String;
Devuelve el prefijo de agregación asociado al cubo actual.

procedure setPrefijoDeAgregacion(strPrefijoDeAgregacion: **in** String);
Cambia el prefijo de agregación, asociado al cubo actual, por strPrefijoDeAgregacion.

function getModoDeAlmacenamiento(): Enumerado;
Devuelve una constante enumerada que identifica el modo de almacenamiento asociado al cubo actual.

procedure setModoDeAlmacenamiento(enumModoDeAlmacenamiento:
Enumerado);
Cambia el modo de almacenamiento, asociado al cubo actual, por enumModoDeAlmacenamiento.

function getFiltroTablaDeHechos(): String;
Devuelve la condición de filtro sobre los registros de la tabla de hechos asociada al cubo actual (condición WHERE de una sentencia SQL).

procedure setFiltroTablaDeHechos(strFiltroTablaDeHechos: **in** String);
Cambia la condición de filtro sobre los registros de la tabla de hechos asociada al cubo actual por strFiltroTablaDeHechos.

function getClausulaFrom(): String;
Devuelve los nombres de las tablas utilizadas en el cubo actual.

procedure setClausulaFrom(strClausulaFrom: **in** String);
Cambia los nombres de las tablas utilizadas en el cubo actual por strClausulaFrom.

function existenAgregaciones(): Boolean;
Indica si existen agregaciones en el cubo actual.

procedure eliminarAgregaciones();
Elimina las agregaciones de todas las particiones existentes en el cubo actual.

function clonar(): CuboActual;
Devuelve una referencia a una copia del objeto CuboActual.

procedure destruir();
Prepara al objeto para liberar la memoria y recursos que consume.

attributes *Atributos relevantes.*

mOrigenDeDatos: OrigenDeDatos;
mBaseDeDatos: BaseDeDatos;
Referencias a objetos.

mComandos: Comandos;
mDimensionesCubo: DimensionesCubo;
mMedidas: Medidas;
mRoles: Roles;
mParticiones: Particiones;
mNombresCubosBase: ColString;
Colecciones.

mstrClave: String;
mstrNombre: String;
mstrDescripcion: String;
mstrTablaDeHechos: String;
mintFilasEstimadas: Integer;
mblnEsRegular: Boolean;
mstrClausulaJoin: String;
mstrPrefijoDeAgregacion: String;
menumModoDeAlmacenamiento: Enumerado;
mstrFiltroTablaDeHechos: String;
mstrClausulaFrom: String;
Otras propiedades.

end CuboActual

Generic class Colección (T);
exports

function AddNew(elem: **in** T): elem;
Agrega una COPIA a la colección del elemento referenciado por el parámetro.

function Item(strIndexKey: **in** String): elem
Devuelve una referencia a un elemento de la colección.
Se puede acceder al elemento deseado por índice (Integer) o por clave (String).

function Count(): Integer;
Devuelve el número de elementos de la colección.

procedure Remove(strIndexKey: **in** String)
Elimina un elemento de la colección.
Se puede eliminar el elemento deseado por índice (Integer) o por clave (String).

procedure destruir();
Prepara al objeto para liberar la memoria y recursos que consume.

procedure setPadre(nuevoPadre: **in** ObjetoPadre);
El valor de la propiedad sólo se puede establecer una vez.
end Coleccion

DimensionesCubo is Coleccion(DimensionCubo);
NivelesCubo is Coleccion(NivelCubo);
PropiedadesMiembro is Coleccion(PropiedadMiembro);
Rebanadas is Coleccion(Rebanada);
Medidas is Coleccion(Medida);
Comandos is Coleccion(Comando);
Roles is Coleccion(Rol);
Particiones is Coleccion(Particion);
Agregaciones is Coleccion(Agregacion);
DimensionesAgregacion is Coleccion(DimensionAgregacion);
NivelesAgregacion is Coleccion(NivelAgregacion);

Los elementos que conforman estas colecciones son los correspondientes a los objetos homónimos de DSO (donde en este modelo de objetos aparecen los nombres de los mismos en Inglés), conteniendo las propiedades que estos proveen, conjuntamente con las funciones Get y Set necesarias para su manipulación. También proveen métodos particulares para simplificar la programación y mantener la consistencia del modelo.

Dado lo extensa que resultaría esta sección al citarlos todos, se cree conveniente omitir su análisis detallado, ya que no contribuiría con grandes aportes. El diseño presentado en este capítulo, en conjunto con el código documentado adecuadamente, y las facilidades que presenta Visual Basic para navegar entre los distintas clases y módulos, resulta adecuado para que un programador pueda entender la totalidad del sistema sin mucha dificultad.

CAPÍTULO 5

Implementación

5. Implementación

5.1 Introducción

En el presente capítulo se realiza un análisis de los distintos módulos de los que se compone la herramienta desarrollada.

En el capítulo anterior se llevó a cabo una presentación de los mencionados módulos, dejando en claro el papel que los mismos juegan en el diseño global de la aplicación. Las siguientes secciones ahondarán en la descripción de cada uno de ellos, exponiendo las herramientas con que interactúan y la forma en que llevan a cabo dicha interacción.

Para cada módulo también se hará mención de algunos detalles de implementación así como de las funcionalidades que otorga.

5.2 Descripción detallada de los módulos

5.2.1 Módulo CargadorDeMetadataOS

Como se mencionó en el cuarto capítulo este módulo es el encargado de realizar la interacción con Microsoft OLAP Services, y hace uso del modelo de objetos DSO para llevar a cabo la comunicación con el mencionado servidor OLAP.

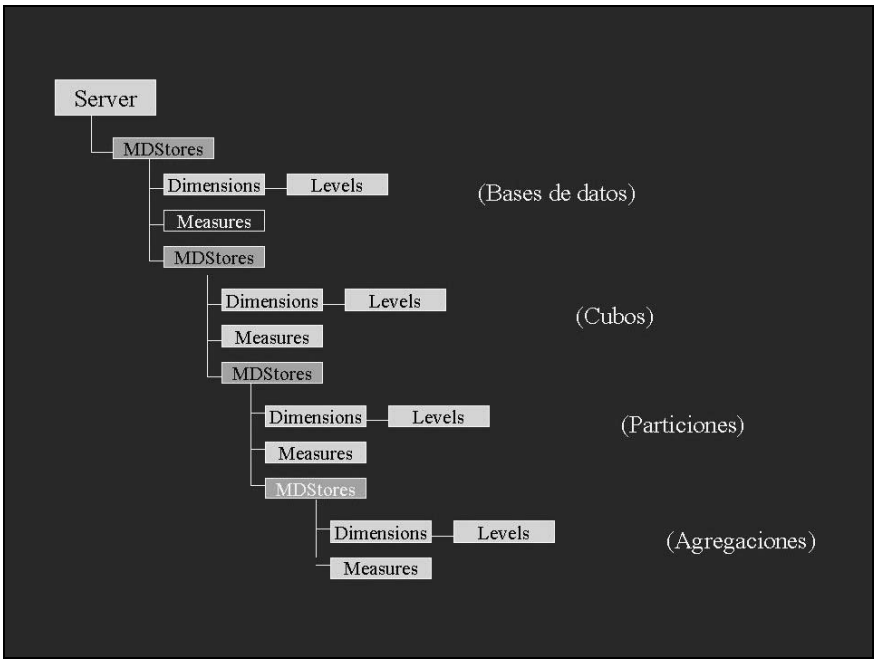
A continuación se expone el modelo de objetos DSO para luego pasar a tratar el tema de la interacción con el mismo.

5.2.1.1 Modelo de Objetos DSO

El modelo de objetos DSO de Microsoft OLAP Services habilita crear aplicaciones que trabajen sobre dicho servidor OLAP. Estas aplicaciones pueden usar DSO para controlar el servidor, crear y mantener objetos OLAP entre otras actividades.

DSO organiza los objetos que componen el modelo en una estructura jerárquica. Estos objetos definen los elementos de las estructuras multidimensionales de OLAP.

La estructura mencionada se muestra en el siguiente diagrama:



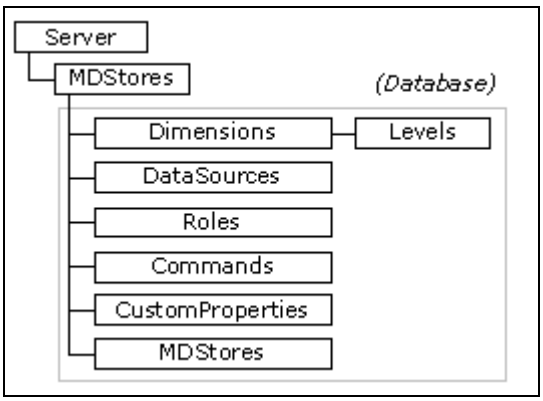
El objeto servidor contiene grupos de objetos que definen bases de datos accesibles desde el servidor. Cada elemento base de datos puede contener grupos de objetos que definen cubos regulares o cubo virtuales. Un cubo contiene una o más particiones, las cuales contienen agregaciones.

Cada objeto que aparece en el diagrama contiene colecciones y objetos. Una colección especial es la llamada *MDStores*, que puede mantener colecciones de bases de datos, cubos (regulares o virtuales), particiones o agregaciones. Los elementos en una colección *MDStores* son identificados por el valor en su propiedad *ClassType*.

Objeto Base de Datos

Una base de datos en DSO contiene las siguientes colecciones: *Dimensions* (con las subordinadas colecciones *Levels*), *DataSources*, *Roles*, *Commands*, *CustomProperties* y una colección *MDStores* de cubos virtuales y regulares. El objeto *Database* se encuentra en una colección *MDStores* bajo el objeto *Server*.

La organización mencionada aparece en el siguiente diagrama:



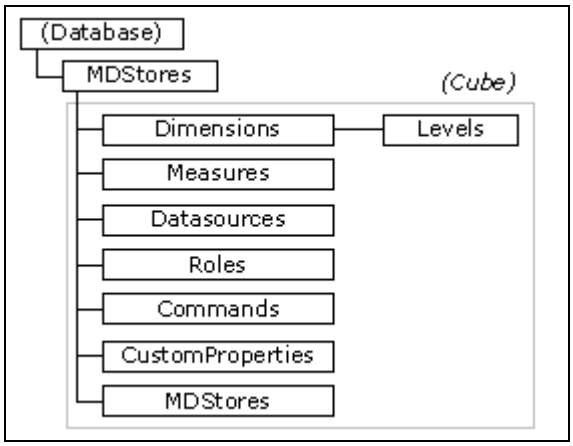
Una descripción de estas colecciones aparece a continuación:

- *Dimensions* contiene las dimensiones privadas y compartidas junto con sus colecciones *Levels*.
- *Datasources* contiene los orígenes de datos aplicables a los objetos subordinados a la base de datos.
- *Roles* contiene los objetos *Role* que especifican los derechos de acceso para los usuarios autorizados.
- *Commands* se reserva para uso futuro.
- *CustomProperties* contiene los objetos *Property* que pueden ser usados para definir propiedades para uso del usuario.
- *MDStores* puede contener cubos regulares o virtuales.

Objeto Cubo Regular

Un cubo regular en DSO contiene las siguientes colecciones: *Dimensions* (con las subordinadas colecciones *Levels*), *Measures*, *DataSources*, *Roles*, *Commands*, *CustomProperties* y una colección *MDStores* que contiene una o más particiones.

La organización mencionada aparece en el siguiente diagrama:



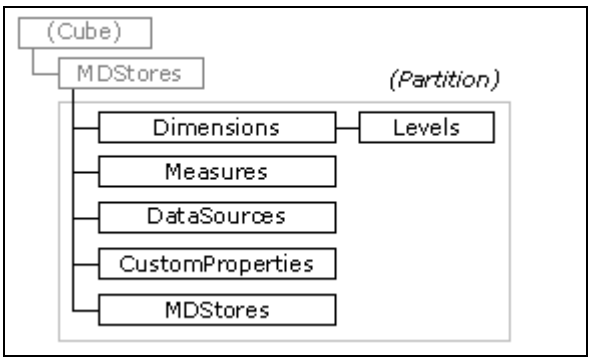
Una descripción de estas colecciones aparece a continuación:

- *Dimensions* contiene las dimensiones usadas por el cubo.
- *Measures* contiene las medidas asociadas al cubo.
- *Datasources* contiene el origen de datos del cubo. Un cubo puede tener un solo origen de datos.
- *Roles* contiene los derechos de acceso al cubo para los distintos usuarios.
- *Commands* contiene expresiones MDX que son empleadas para especificar información utilizada en el cubo.
- *CustomProperties* contiene los objetos *Property* que pueden ser usados para definir propiedades útiles al usuario.
- *MDStores* contiene una o más particiones.

Objeto Partición

Una partición en DSO contiene las siguientes colecciones: *Dimensions* (con las subordinadas colecciones *Levels*), *Measures*, *DataSources*, *CustomProperties* y una colección *MDStores* que contiene agregaciones.
Cada cubo contiene por lo menos una partición que es su partición primaria.

La organización mencionada aparece en el siguiente diagrama:



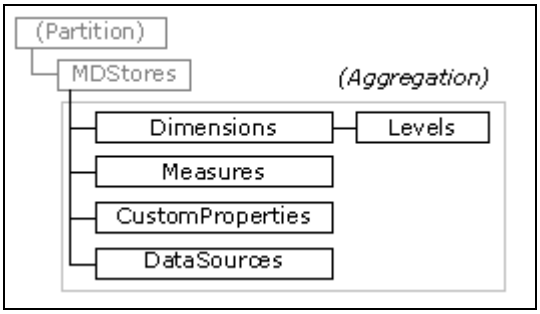
Una descripción de estas colecciones aparece a continuación:

- *Dimensions* contiene las dimensiones de la partición.
- *Measures* contiene las medidas para la partición.
- *Datasources* contiene el origen de datos de la partición.
- *CustomProperties* contiene los objetos *Property* que pueden ser usados para definir propiedades útiles al usuario.
- *MDStores* contiene las agregaciones asociadas a la partición.

Objeto Agregación

Una agregación en DSO contiene las siguientes colecciones: *Dimensions* (con las subordinadas colecciones *Levels*), *Measures*, *DataSources* y *CustomProperties*.

La organización mencionada aparece en el siguiente diagrama:



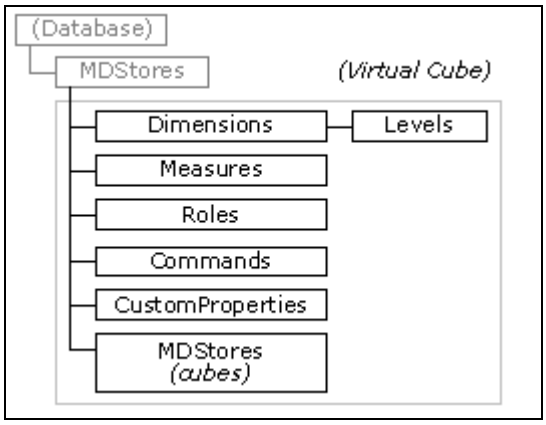
Una descripción de estas colecciones aparece a continuación:

- *Dimensions* contiene las dimensiones usadas por la agregación.
- *Measures* contiene las medidas usadas en la agregación.
- *Datasources* contiene los orígenes de los datos de la agregación.
- *CustomProperties* contiene los objetos *Property* que pueden ser usados para definir propiedades útiles al usuario.

▪ Objeto Cubo Virtual

Un cubo virtual en DSO contiene las siguientes colecciones: *Dimensions* (con las subordinadas colecciones *Levels*), *Measures*, *Roles*, *Commands*, *CustomProperties* y una colección *MDStores* que contiene los cubos base del cubo virtual correspondiente.

La organización mencionada aparece en el siguiente diagrama:



Una descripción de estas colecciones aparece a continuación:

- *Dimensions* contiene las dimensiones usadas por el cubo virtual.
- *Measures* contiene las medidas asociadas al cubo virtual.
- *Roles* contiene los objetos *Role* que especifican los derechos de acceso para los usuarios autorizados.
- *Commands* contiene expresiones MDX que son usadas para especificar información usada en el cubo.
- *CustomProperties* contiene los objetos *Property* que pueden ser usados para definir propiedades útiles al usuario.
- *MDStores* contiene los cubos base asociados al cubo virtual.

5.2.1.2 Interacción con DSO

Las funcionalidades que provee el módulo que se está describiendo pueden agruparse en tres grupos:

- las que manejan las conexiones de la aplicación con el servidor OLAP
- las que recorren la estructura de DSO para extraer objetos del servidor OLAP
- las que introducen nuevos elementos al servidor OLAP

El manejo de las conexiones se realiza desde el objeto *Server*. Este objeto proporciona las funciones *Connect* y *CloseServer*, que son las encargadas de establecer y cerrar la conexión con el servidor OLAP respectivamente. La invocación a las mencionadas funciones se realiza de la siguiente forma:

```
DsoServer.Connect "Nombre_de_servidor" , y

DsoServer.CloseServer
```

Donde *DsoServer* es una instancia del objeto *Server*.

Para el segundo grupo de funciones, que son las que extraen datos del servidor OLAP, lo que se hizo fue recorrer el modelo de objetos citado en la sección anterior. Para tal fin, se exploró la jerarquía del árbol que representa la estructura, desde el nodo de nivel superior hasta llegar a los de nivel inferior.

De manera de permitir este tipo de acciones, las colecciones de DSO cuentan con funciones de tipo similar a las que proporcionan las colecciones de Visual Basic para posibilitar su recorrido.

Los objetos que introduce al servidor OLAP el presente módulo son: *Datasources*, *Databases* y *Cube*. Al introducir el último objeto mencionado también se agregan todos los elementos subordinados a éste (*Dimensions*, *Measures*, etc.).

Los distintos objetos son extraídos de un objeto de tipo *CuboActual*, que es la estructura que utilizamos para almacenar temporalmente los elementos de la estructura multidimensional, que se describirá más adelante en este mismo capítulo.

Luego de introducir los objetos en el servidor, para que los cambios en el repositorio de metadata se tomen en cuenta, se debió invocar para algunos de los objetos que utilizan la interface *MDStore* la función *Update* .Los cambios no son considerados por el servidor OLAP hasta la invocación de la citada función.

Estrechamente ligado al *CargadorDeMetadataOS* se encuentra el formulario *frmProcesarCubo*, el cual se encarga del procesamiento del cubo recién creado. Para llevar a cabo tal tarea se utiliza el método *Process* del objeto *Cube*. Dicho método se encuentra definido en la interface *MDStore* y se encarga de crear y poblar un objeto *MDStore* en un

servidor OLAP. Bases de datos, cubos y particiones pueden ser procesados. Procesar cada uno de estos objetos hace que todos sus objetos subordinados sean procesados. Por ejemplo, invocando el método *Process* de una base de datos se procesan todos los cubos y dimensiones definidos sobre la base de datos.

Cuando se procesa un cubo, se calculan los agregados diseñados para el cubo y se carga el cubo con los agregados y datos calculados. Procesar un cubo implica leer las tablas de dimensiones para llenar los niveles con miembros de los datos actuales, leer la tabla de hechos, calcular los agregados específicos y almacenar los resultados en el cubo. Una vez que se ha procesado un cubo, pueden consultarlo los usuarios.

5.2.2 Módulo CargadorDeDatos

Este módulo es el encargado de la transferencia de las bases de datos en que se basan los cubos de OLAP Services, desde el servidor origen hasta el destino seleccionado. Para llevar a cabo la tarea descrita se hace uso de los paquetes de servicios de transformación de datos (DTS), que fueron presentados en capítulos previos.

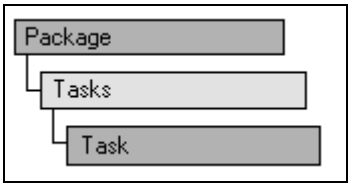
En la sección siguiente se describirán más detalladamente estos paquetes, para luego pasar a comentar la forma en que fueron utilizados.

5.2.2.1 Paquetes DTS

Los paquetes de los servicios de transformación de datos incluyen las siguientes colecciones:

- Conexiones que definen cada origen de datos (origen y destino).
- Tareas que definen las acciones que se van a realizar.
- Pasos que definen la secuencia en que se van a realizar las tareas.

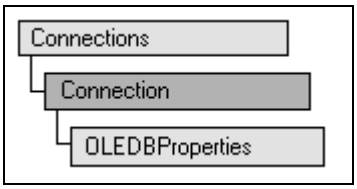
- **Colección de tareas**



La colección *Tasks* es un grupo de objetos *Task* que contiene información acerca de las unidades de trabajo a ser realizado como parte de un proceso de transformación de datos. La colección de tareas contiene todas las tareas del correspondiente paquete DTS.

Las tareas de las que se hizo uso en el *CargadorDeDatos* son las de transferencia de objetos DTS. Estas tareas proporcionan un eficaz mecanismo para transferir objetos y datos de bases de datos de una base de datos de Microsoft SQL Server a otra. Se puede transferir todos los objetos de una base de datos o sólo un subconjunto que se selecciona cuando se define la tarea. Si sólo se transfieren objetos seleccionados, la tarea puede especificar que todos los objetos dependientes también se transfieran.

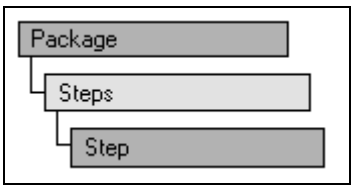
- **Colección de conexiones**



Los objetos *Connection* contienen información acerca de conexiones con proveedores de servicios OLE DB.

Las conexiones utilizadas en el módulo que se está describiendo son conexiones de orígenes de datos. La información que incluyen estos objetos abarca nombres de servidores, el formato y ubicación de los datos, y las contraseñas. La primera tarea que utilice la conexión establece dicha conexión.

- **Colección de pasos**



La colección *Steps* es un grupo de objetos *Step* que contienen información acerca del flujo y ejecución de tareas de un paquete DTS.

Una tarea que no tiene un objeto de paso asociado nunca se ejecuta. Los objetos de paso proporcionan funcionalidades de "flujo de trabajo" al paquete para que las tareas se ejecuten en una secuencia determinada. Cada paso se ejecuta cuando se han satisfecho todas sus restricciones de precedencia. Si un paso no define ninguna restricción de precedencia, es apto para su ejecución inmediata. Los pasos subsiguientes pueden reutilizar las tareas, pero sólo pueden hacerlo de uno en uno.

5.2.2.2 Interacción con los paquetes DTS

Las funcionalidades principales proporcionadas por el módulo que se está analizando son las de:

- crear un paquete de transformación de datos, y
- ejecutar el paquete creado

Con el fin de crear un paquete DTS se realizaron los siguientes pasos:

1. Se creó un objeto *Package*.
2. Se creó una colección de conexiones para el paquete.
3. Se agregaron objetos *Connection* a la colección creada en el paso anterior.
4. Se creó una colección de tareas.
5. Se agregaron objetos *Task* a la colección creada en el paso anterior.
6. Se creó una colección de pasos.
7. Se agregaron objetos *Step* a la colección de pasos.

El objeto tarea utilizado fue *TransferObjectsTask*. Esta tarea permite transferir uno o más objetos Microsoft SQL Server entre un servidor origen y un destino. Un objeto puede representar: una tabla, una vista, un procedimiento almacenado, un índice, etc.
En el caso de la implementación realizada se utilizó la tarea mencionada para que se transfirieran todos los objetos de una base de datos origen ubicada en un servidor, hacia una destino ubicada en otro servidor.

La ejecución del paquete se lleva a cabo mediante el método *Execute* del objeto *Package*. Previamente a la ejecución del paquete, el procedimiento encargado de la ejecución crea una base de datos en el servidor destino, para luego completar esta base con la ejecución del paquete DTS. La creación de la mencionada base se realiza utilizando el comando:

```
CREATE DATABASE “Nombre_Bdatos”
```

Donde *Nombre_Bdatos* es el nombre que se le quiere dar a la base de datos destino.
Este comando es enviado a SQL Server para su ejecución haciendo uso del objeto *Command* proporcionado por ADO.

En el paquete DTS se especifican nombres de usuario y contraseñas para poder establecer la comunicación con los distintos servidores SQL Server. Si alguno de estos datos no es correcto la tarea no se va a poder llevar a cabo.

5.2.3 Módulo MMI

La mención de este módulo está abarcando los distintos formularios que constituyen la interfaz de la aplicación, así como los módulos muy relacionados a ésta que contribuyen con su tarea.

La interfaz cuenta como elementos principales con un menú, una barra de tareas y los objetos de Visual Basic *TreeView* y *ListView*. En estos dos últimos elementos es donde se puede visualizar la metadata del cubo con que se está trabajando.
Para los distintos elementos de la metadata de un cubo se cuenta con formularios específicos, que permiten observar sus propiedades así como realizar algunas modificaciones sobre las mismas.

El módulo que se está analizando permite trabajar sobre varios cubos al mismo tiempo. Para tal fin se mantiene una estructura de datos para contener los cubos abiertos en un momento dado. La estructura mencionada consiste un una colección de elementos de tipo *Registro*.

Un objeto *Registro* contiene los siguientes elementos:

- Un objeto de tipo *CuboActual*, donde se almacena la metadata de un cubo.
- Una especificación del tipo de origen del cubo, mencionando si se trata de un cubo que ya ha sido exportado o no.
- Una mención acerca del lugar de origen del cubo, que es un servidor OLAP en el caso de que sea un cubo que todavía no se ha exportado, o la ruta absoluta a un lugar de almacenamiento en disco en el caso de que se trate de un cubo que ha sido exportado.
- En el caso de que el cubo sea virtual y este se halla exportado junto a sus cubos base, también se almacenan las rutas al lugar en disco donde estos están almacenados.

Mediante la interfaz proporcionada, el usuario puede observar la metadata de un determinado cubo, realizar las modificaciones admitidas sobre sus elementos, llevar a cabo exportaciones e importaciones, así como el resto de las funcionalidades que proporciona la aplicación.

El presente módulo es el encargado de imponer restricciones acerca de las modificaciones que el usuario se ve en libertad de realizar sobre un determinado cubo. Entre estas restricciones se encuentran las siguientes:

- No permitir que un cubo se quede sin dimensiones o sin medidas como producto de eliminaciones sucesivas.
- No permitir que una dimensión se quede sin niveles como consecuencia de sucesivas eliminaciones.

- Restringir el largo de los nombres y las descripciones de los elementos.
- Restringir los caracteres que pueden contener los nombres de los elementos.

Además, como se comentó en el capítulo destinado al análisis, solo un subconjunto de las propiedades de los objetos es libre a ser modificado.

A pesar de las restricciones impuestas, ciertas operaciones pueden dejar inválidos elementos de la metadata del cubo. Como ejemplos de este hecho cabe mencionar las eliminaciones de dimensiones, niveles o propiedades miembro, que pueden llegar a dejar inválidos miembros calculados.

5.2.4 Módulo CargadorDeMetadataExp

Para trabajar con XML, se debe tener algún medio para acceder a los datos. Un módulo de software capaz de leer documentos XML y proporcionar acceso a su contenido y estructura es conocido como API XML, procesador XML o parser XML (entre otros).

Mientras que los desarrolladores son libres de implementar su propio software para procesar el contenido de un documento XML (que después de todo, es un archivo de texto), lo recomendable es apegarse a los "estándares ya aceptados". Adoptando estos últimos, es posible escribir código para una implementación particular de API *(Application Program Interface)* que debería tener la capacidad de correr bajo cualquier otra implementación de la misma API sin modificaciones.

Existen dos especificaciones de interfaces de programación principales, que han ganado popularidad en el mundo XML, y compiten por convertirse en verdaderos estándares: DOM *(Document Object Model)* y SAX *(Simple API for XML)*.

Esta sección, demuestra como se puede accesar un documento XML utilizando la interfaz de uso común proporcionada por DOM, tal cual es expuesta en la implementación del procesador XML definida por Microsoft (MSXML.DLL), ya que fue la elegida para implantar los métodos que provee la clase *CargadorDeMetadataExp*.

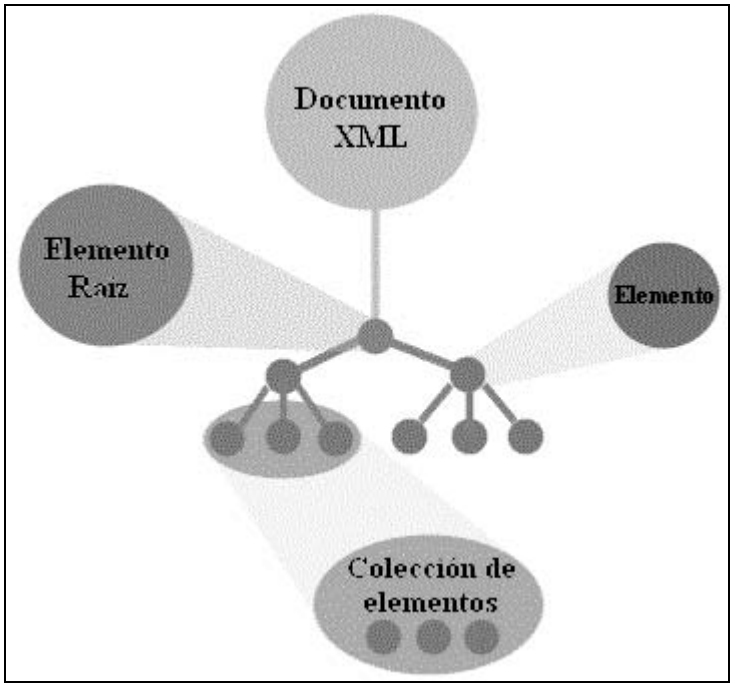
Es recomendable, consultar primero el Apéndice B para familiarizarse con el entorno XML, así como las facilidades y ventajas que este lenguaje proporciona.

5.2.4.1 Modelo de objetos DOM

Como se mencionó en el capítulo previo, con el modelo de objetos de documento, es posible cargar y analizar archivos XML.

DOM expone el documento XML como una estructura de árbol compuesta por nodos; y provee de interfaces de programación adecuadas para recorrer este modelo y manipularlo.

Cada nodo es definido con un tipo específico que restringe las relaciones de parentesco que puede tener con otros nodos (padre, hijo, etc.). Para la mayoría de los documentos XML, los tipos de nodos más comunes son elementos, atributos y texto. Los atributos ocupan un lugar especial en el modelo, ya que no son considerados hijos de algún padre, por esta razón se suministra una interface de programación particular para trabajar con ellos.



Los cuatro objetos principales, expuestos en la implementación de XML DOM proporcionada por Microsoft, son:

- XMLDOMDocument
- XMLDOMNode
- XMLDOMNodeList
- XMLDOMNamedNodeMap
- XMLDOMParseError

Cada uno de estos objetos revela métodos y propiedades que permiten reunir información sobre la instancia del objeto, manipular el valor y estructura del mismo, y navegar por otros objetos dentro del árbol.

Para programadores en Microsoft Visual Basic, estos son expuestos como las interfaces COM: *IXMLDOMDocument, IXMLDOMNode, IXMLDOMNodeList, IXMLDOMNamedNodeMap,* e *IXMLDOMParseError* respectivamente.

A continuación se incluye una muy breve explicación de cómo se utiliza el modelo descripto, y los métodos y propiedades relevantes que proporciona.

- **XMLDOMDocument**

Este objeto puede ser considerado el más importante, ya que provee propiedades y métodos que permiten navegar, consultar y modificar el contenido y estructura de un documento XML.

La utilización del modelo, comienza con la creación del objeto *XMLDOMDocument.* Una vez obtenido un objeto documento, se puede comenzar a cargar, parsear, navegar y manipular archivos XML.

Para desarrolladores en Microsoft Visual Basic, un nuevo objeto *XMLDOMDocument* vacío, puede ser creado de la siguiente manera:

```
Dim xmlDoc As MSXML.DOMDocument
Set xmlDoc = New MSXML.DOMDocument
```

Mediante la referencia obtenida previamente, puede ser invocado el método *load* para cargar un archivo XML a partir de su ruta (absoluta o relativa), URL, IIS *(Internet Information Server)* Request, etc. El siguiente ejemplo, demuestra como cargar un archivo XML dada su ruta absoluta:

```
xmlDoc.load( "c:\Mis documentos\Sales.xml")
```

El objeto documento, xmlDoc, ahora contiene un árbol consistente en el contenido parseado de Sales.xml.

Otra alternativa, es cargar la estructura de documento a partir de una cadena, utilizando el método *loadXML:*

```
xmlDoc.loadXML("<CuboActual><Nombre>Sales</Nombre></CuboActual>")
```

Finalmente, para almacenar un documento XML en un archivo, se debe invocar al método *save* proporcionando el destino deseado:

```
xmlDoc.save("c:\Mis Documentos\nuevoSales.xml")
```

El objeto *XMLDOMDocument* expone propiedades que permiten al programador cambiar el comportamiento de análisis del documento *(parsing)* en tiempo de ejecución. Estas cuatro propiedades son *async, validateOnParse, resolveExternals,* y *preserveWhiteSpace.* Cada una de ellas toma y retorna un valor Booleano (siendo True el valor por defecto para las tres primeras).

El ejemplo que se incluye a continuación, carga y parsea un archivo XML en forma sincrónica; sin embargo, el documento no es validado durante la carga:

```
xmlDoc.async = False
xmlDoc.validateOnParse = False
xmlDoc.load("ejemplo.xml")
```

El análisis del documento en forma asincrónica, permite a las aplicaciones hacer otras cosas (como desplegar un indicador de progreso) mientras el documento es cargado. En estos casos, antes de comenzar a trabajar con el modelo, se debe consultar la propiedad *readyState* para asegurarse de que el mismo ha sido completamente parseado y que DOM está disponible.

Existen muchas otras propiedades que permiten obtener información concerniente al documento: *doctype, implementation, url,* etc., de las que se omite su comentario.

XMLDOMDocument permite accesar la estructura de árbol, ya sea partiendo del elemento raíz y recorriendo la estructura en profundidad, o a través de consultas que retornen un conjunto de nodos específicos. El fragmento de código que se incluye a continuación, despliega el texto de cada hijo directo de la raíz:

```
Dim xmlDoc As MSXML.DOMDocument
Dim xmlRaiz As MSXML.IXMLDOMNode
Dim xmlHijo As MSXML.IXMLDOMNode

Set xmlDoc = New MSXML.DOMDocument
xmlDoc.async = False
xmlDoc.load("c:\ejemplo.xml")

Set xmlRaiz = xmlDoc.documentElement
For Each xmlHijo In xmlRaiz.childNodes
      MsgBox xmlHijo.text
Next xmlHijo
```

Es posible navegar hacia un nodo determinado, o hacia un conjunto de nodos de la estructura, utilizando los métodos *nodeFromID* o *getElementsByTagName.* El primero de ellos toma un ID único, definido dentro de un Schema XML o DTD, y retorna el nodo correspondiente a ese ID específico. El segundo método, toma una cadena que representa una marca determinada y retorna todos los nodos coincidentes.

- **XMLDOMNode**

Una vez que el documento ha sido cargado, el siguiente paso es obtener la información de interés contenida en su estructura. Si bien el objeto documento es importante, la interface *IXMLDOMNode* es la utilizada con mayor frecuencia en las aplicaciones, ya que proporciona rutinas para leer y escribir nodos individuales.

Antes de proseguir, es interesante comentar que existen actualmente trece tipos de nodos soportados por el procesador MSXML. La siguiente tabla, lista unos pocos de los más importantes, junto con algunos ejemplos ilustrativos:

DOM NodeType	Ejemplo
NODE_ELEMENT	**<Comando** Tipo="cmdCreateMember">Profit**</Comando>**
NODE_ATTRIBUTE	<Comando **Tipo="cmdCreateMember"**>Profit</Comando>
NODE_TEXT	<Comando Tipo="cmdCreateMember">**Profit**</Comando>
NODE_PROCESSING_INSTRUCTION	**<?xml version="1.0"?>**
NODE_DOCUMENT_TYPE	**<!DOCTYPE ejemplo SYSTEM "ej.dtd">**

El tipo de nodo puede ser accedido a través de dos propiedades principales: *nodeType* y *nodeTypeString*. La primera expone un tipo de datos enumerado, que indica la naturaleza del nodo (algunas de estas constantes enumeradas, se listaron en la tabla previa), mientras que la segunda permite obtener una representación textual del tipo de nodo.

Una vez obtenida una referencia a un objeto documento, se puede comenzar a recorrer el modelo jerárquico. Desde esta referencia, accesando la propiedad *childNodes,* se obtiene un punto de entrada a todos los nodos que conforman el documento XML. Esta propiedad soporta el constructor de Visual Basic For Each Next, lo que resulta extremadamente útil al momento de recorrer sus elementos.

No solo el objeto documento proporciona la propiedad *childNodes*, sino también la exponen todos los nodos individuales, de esta manera, en conjunción con la propiedad *hasChildNodes,* es posible navegar la estructura de elementos, atributos y valores del documento XML.

Como el modelo proporcionado por DOM es jerárquico por naturaleza, resulta relativamente sencillo escribir una rutina recursiva que recorra el documento en su totalidad.

- **XMLDOMNodeList**

El objeto *XMLDOMNodeList* es retornado por la propiedad *childNodes*, y los métodos *selectNodes* y *getElementsByTagName*. Consiste simplemente en una lista ordenada de nodos, que puede ser recorrida y consultada con facilidad, ya que proporciona escasos dos métodos *(item* y *nextNode)* y una única propiedad que indica el largo de la lista.

- **XMLDOMNamedNodeMap.**

El objeto *XMLDOMNamedNodeMap* es devuelto por la propiedad *attributes*, y a diferencia del objeto comentado previamente, esta colección de nodos (atributos) puede ser accesada por nombre.

XMLDOMNamedNodeMap proporciona todos los métodos y propiedades que expone *XMLDOMNodeList*, sin embargo, permite interactuar con sus miembros a través de los métodos adicionales *getNamedItem* y *getQualifiedItem.* El primero de ellos toma el nombre del atributo deseado como parámetro, mientras que el segundo, además, recibe el namespaceURI. Ambos retornan un objeto *XMLDOMNode.*

Finalmente, existen tres métodos para manipular los miembros de *XMLDOMNamedNodeMap: setNamedItem, removeNamedItem,* y *removeQualifiedItem.* Estos son utilizados para agregar y eliminar elementos de la colección de atributos.

- **XMLDOMParseError**

Un documento XML, puede fallar durante el proceso de carga por múltiples razones. Una causa común, puede ser que la ruta de archivo proporcionada al método *load* sea inválida. Otra causa puede ser que el documento XML, en sí mismo, sea inválido.

Por defecto, MSXML valida los documentos contra el DDT o Schema que haya sido especificado. De todos modos, es posible indicarle al parser que no efectúe la validación, cambiando la propiedad *validateOnParse* del objeto *DOMDocument* antes de invocar al método *load.*

Independientemente del tipo de falla, la información concerniente al origen y descripción del error, puede ser solicitada al procesador XML accesando el objeto *XMLDOMParseError.* Este objeto, expone siete propiedades de gran utilidad a la hora de investigar la causa del problema. Esta información, posteriormente puede ser desplegada al usuario, almacenada en un archivo de registro o utilizada para corregir la anomalía mediante código.

5.2.4.2 Interacción con DOM

Lo anterior es solo una versión muy resumida, del modelo de objetos que implementa Microsoft en su versión 2.0 del procesador MSXML.DLL.

El parser MSMLX.DLL (que es un componente integral de Microsoft Internet Explorer 5), implementa todas las interfaces (fundamentales y extendidas) recomendadas por el W3C *(World Wide Web Consortium)* en octubre de 1998. También provee otros métodos para soportar XSL *(Extensible Stylesheet Language)*, XSL Patterns, namespaces, data types, etc.

El módulo *CargadorDeMetadataExp* comenzó interactuando con la versión 1.0 de este parser, y posteriormente migró a la versión 2.0 (última hasta ese momento). En la actualidad existen otras dos versiones: 2.6 y 3.0.
Es de destacar, que todas estas versiones están en desarrollo actualmente; por lo que presentan inconsistencias y errores en su implementación. Del mismo modo revelan importantes carencias y omisiones en la documentación que se proporciona sobre su interfaz de programación.

La clase *CargadorDeMetadataExp* provee de tres métodos: *generarCuboExp*, *generarCuboActual*, y *transformar*, que se detallan a continuación:

- *generarCuboExp*

Este método permite crear una instancia XML, a partir de la estructura multidimensional que mantiene nuestra aplicación *(CuboActual)*. Es de señalar el hecho de que la instancia generada es válida, es decir, la misma se declara conforme a un *Schema* XML y se ajusta positivamente al mismo.
Este *Schema* XML, fue desarrollado con el propósito de suministrar a otras herramientas, la lista de tipos de elementos disponibles y las restricciones impuestas en la estructura del documento generado.

Para especificar el esquema de documento, se utilizó el formato XDR *(XML Data Reduced Schema)*, que tiene características que lo hacen adecuado a nuestra problemática. En la actualidad, existen cerca de siete formatos básicos de esquemas, cada uno con sus propias ventajas, limitaciones y sintaxis. No es el propósito de esta sección describir el formalismo necesario para escribir esquemas, por lo que solo se comentarán las ventajas inmediatas encontradas al formato XDR utilizado:

- Sintaxis menos compleja.
- Ofrece utilidades semánticas más eficaces al incorporar el concepto de relaciones (0:1, 0:n, 1:1, 1:n) y herencia entre sus elementos.
- Permite especificar tipos de datos comunes o enriquecidos, enumerados, etc.
- Permite plasmar nuestra estructura multidimensional *(CuboActual)*, con facilidad.

El esquema desarrollado se proporciona en un archivo de nombre “Cubo.xdr”, en cada proceso de exportación efectuado en nuestra herramienta.

- *generarCuboActual*

Su finalidad es exactamente la opuesta al método anterior, es decir, su trabajo consiste en cargar nuestro modelo de cubo *(CuboActual)* a partir de la información contenida en una instancia XML válida.

- *transformar*

Permite aplicar una hoja de estilo *(XSL, Extensible Stylesheet Language)* a una instancia XML determinada.

Este método es un componente fundamental a la hora de permitir visualizar la metadata de un cubo en un browser. Para tal fin, se desarrollaron dos hojas de estilo *(Regular.xsl, Virtual.xsl)*, las cuales al ser aplicadas (a cubos regulares o virtuales respectivamente) generan documentos HTM.

En la subcarpeta Recursos instalada con la aplicación, se distribuyen los archivos ya mencionados:

Cubo.xdr
Esquema conforme al formato XML Data Reduced.

Cubo.xsl
Hoja de estilo que permite visualizar la estructura del esquema (Cubo.xdr) con mayor claridad.

Regular.xsl, Virtual.xsl
Hojas de estilo para instancias XML generadas por la herramienta.

Todos estos archivos son compatibles con Microsoft Internet Explorer 5.

5.2.5 Módulo IO (Input/Output)

Una de las principales ventajas de trabajar con XML es su estructura abierta.
Utilizando un editor de textos, es posible modificar cualquier archivo con formato XML y alterar sus datos.

No obstante, en algunos casos esto puede ser un inconveniente, ya que muchas aplicaciones necesitan una determinada consistencia en estos datos para trabajar.
Una forma de remediar este problema, es utilizar un DDT o Schema que imponga restricciones sobre la estructura y datos del documento. Sin embargo, con este mecanismo, el conjunto de restricciones que se pueden establecer es en extremo pobre. Además, siempre existe la posibilidad de alterar el DDT o Schema (que también es un archivo de texto) para que se ajuste a la instancia XML modificada.

En nuestro caso particular, es necesario procesar las instancias XML (que el usuario solicite), para cargar la estructura multidimensional sobre la cual trabaja la herramienta, con el fin de modificarla e importarla a un Servidor OLAP. Como se puede apreciar, la consistencia de los datos a cargar es crítica.

Ante esta eventualidad, las alternativas encontradas son dos:

- Validar únicamente la instancia XML conforme a un Schema (ya que permite especificar un conjunto más elaborado de restricciones que un DDT)
- No cargar instancias (previamente generadas por la aplicación) que hayan sido modificadas

La primera de ellas presenta el inconveniente comentado con anterioridad, por lo que también se implementó la segunda alternativa.

Para ello, una vez efectuada la exportación de un cubo de OLAP Services, se genera un pequeño archivo binario (con extensión .cub) que almacena una estructura denominada *RegExportación.*

Esta estructura básicamente, contiene:

- El nombre y fecha de última modificación del archivo XML, correspondiente al cubo que se acaba de exportar.
- Los nombres y fechas de última modificación de los archivos XML, correspondientes a los cubos base (cuando corresponda)

De esta manera, la aplicación solo abre archivos de extensión .cub con el formato descripto, y compara la información almacenada en el registro de exportación *(RegExportación)*, para determinar la validez de los archivos a los que se hace referencia. Si esta validación resulta exitosa, se procede a la carga de las instancias XML allí indicadas. Con este mecanismo, existe plena seguridad de que las instancias XML que la herramienta carga, no han sido modificadas y corresponden efectivamente a documentos XML válidos generados por la aplicación.

Todo este manejo y validación de archivos, se delimita al módulo *IO*. De este modo, la interface gráfica *(MMI)* puede efectuar, en forma completamente transparente, las operaciones *abrir, guardar y guardar como* sobre el conjunto de objetos *Registro* que administra. A continuación se detalla el propósito de cada uno de estos métodos proporcionados por *IO:*

- *abrir*

Permite abrir un archivo binario .cub, y devolver un objeto *Registro* con toda la información (se aconseja consultar la sección que describe el módulo MMI, ya que enumera los componentes del objeto Registro)

- *guardar*

Guarda el objeto *CuboActual*, que se le indique como parámetro, actualizando el archivo binario (.cub) correspondiente.

- *guardarComo*

Permite guardar la información del objeto *Registro*, pasado como parámetro, bajo la nueva ubicación especificada por el usuario. Si el cubo contenido en este registro, no ha sido exportado aún, esta rutina efectúa la exportación.

El proceso de exportación, dependerá de la naturaleza del cubo. Para cubos regulares, consiste simplemente en almacenar la estructura que se encuentra en memoria, en un archivo con formato XML. Para cubos virtuales, se brinda la opción de exportar también sus cubos vinculados (cubos base). En este último caso, se efectúa una conexión con el Servidor OLAP origen obteniendo cada uno de ellos con el propósito de almacenarlos en archivos XML.

En ambos tipos de exportación (cubos virtuales, cubos regulares), se genera el archivo .cub correspondiente y se copian al destino el esquema XML (Cubo.xdr) y su hoja de estilo (Cubo.xsl).

Como se puede apreciar, estos métodos *(abrir, guardar, guardarComo)* utilizan los servicios que brindan los componentes *CargadorDeMetadataOS* y *CargadorDeMetadataExp* según corresponda.

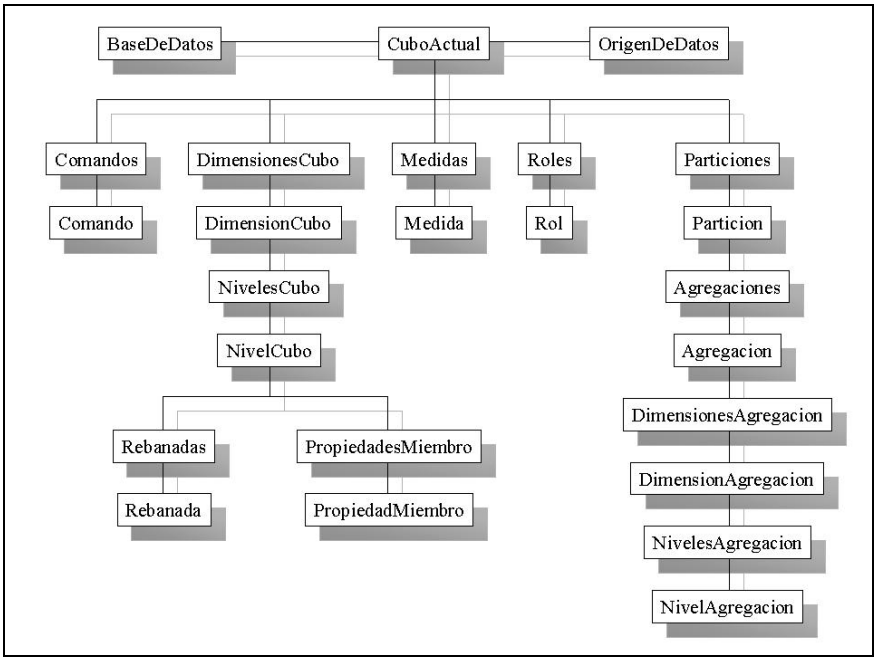
Finalmente, este módulo proporciona el método *generarArchVisualización*, que utilizando la rutina *transformar* (proporcionada por el componente *CargadorDeMetadataExp*) en conjunto con la hoja de estilo que corresponda al tipo de cubo (*Regular.xsl, Virtual.xsl),* genera un documento HTM y lo almacena bajo la ubicación especificada por el usuario.

Este documento HTM, permitirá visualizar la estructura del cubo contenido en el objeto *Registro* (indicado como parámetro), en un browser.

5.2.6 CuboActual

Como se adelantó en el capítulo cuatro, esta estructura tiene como principal cometido almacenar toda la información necesaria para su posterior reconstrucción en el Servidor OLAP. También debe proporcionar los medios para modificar esta información en forma consistente y debe exponer una estructura sencilla para facilitar su representación en un documento XML. En consecuencia, este modelo no debe ser considerado como una propuesta de especificación multidimensional, puesto que está ligado a las restricciones impuestas por DSO y a la forma de representación jerárquica de XML.

Un ejemplo ilustrativo de esta dependencia, lo constituye la existencia de los objetos BaseDeDatos, OrigenDeDatos, Particion, Agregacion, etc., propios de DSO y absolutamente necesarios al momento de reproducir un cubo en OLAP Services.



Desde el punto de vista de programación, todos los objetos (exceptuando el CuboActual) exponen una referencia al padre. De esta manera es posible navegar por la jerarquía en cualquier sentido, desde cualquier objeto. Esto genera referencias circulares entre los componentes de la jerarquía, lo que evita su destrucción. Puesto que Visual Basic no proporciona los medios adecuados para solucionar este inconveniente, todos los objetos exponen un método *destruir* que libera automáticamente las referencias circulares de su agrupación subordinada.

El siguiente ejemplo, libera todos los recursos utilizados por unCubo (de tipo CuboActual) y sus objetos dependientes:

```
unCubo.destruir
Set unCubo = Nothing
```

Finalmente, otro punto relevante en cuanto al manejo de memoria, esta dado por el hecho de la copia automática que se realiza de los elementos que se agregan en alguna colección del modelo. De esta manera se evita compartir memoria y se facilita considerablemente la utilización de la estructura.

DSO identifica los componentes del modelo expuesto, únicamente a través de su nombre. Esto provoca serias inconsistencias al momento de renombrar y eliminar estos elementos en nuestra aplicación, las cuales son subsanadas en su gran mayoría, dejando a criterio del usuario aquellas que conllevan un importante tiempo de cómputo el remediar.

Solo a manera de ejemplo, DSO identifica las dimensiones privadas de un cubo, por el formato que presentan en su nombre:

```
NomCubo^NomDim

/*
Donde NomCubo es el nombre del cubo al cual pertenece la dimensión
privada, siendo esta última expuesta en la consola Administrativa de OLAP
Services como NomDim.
*/
```

Claramente, al renombrar un cubo en nuestra herramienta, se debe mantener la consistencia en el criterio empleado por DSO.

Es importante señalar, que para evitar las restricciones que impone el reconocer los objetos a través de su nombre, los elementos que constituyen nuestra estructura se identifican por claves.

En lo que respecta a la representación del modelo en XML, la idea de diseño utilizada, permite que pueda ser plasmado naturalmente en diferentes niveles de anidamiento. A continuación se incluye un bosquejo de la disposición general que obedecen los elementos, en las instancias XML generadas por el producto.

```
<?xml version="1.0"?>
<!--Generado por Migrador OLAP.-->
<CuboActual xmlns="x-schema:Cubo.xdr">
      <OrigenDeDatos>
            ...
      </OrigenDeDatos>

      <BaseDeDatos>
            ...
      </BaseDeDatos>

      <Comandos>
            <Comando>
               ...
            </Comando>
      </Comandos>

      <DimensionesCubo>
            <DimensionCubo>
                  <NivelesCubo>
                        <NivelCubo>
                              <Rebanadas>
                                    <Rebanada>
                                       ...
                                    </Rebanada>
                              </Rebanadas>

                              <PropiedadesMiembro>
                                    <PropiedadMiembro>
                                          ...
                                    </PropiedadMiembro>
                              </PropiedadesMiembro>
                             ...
                        </NivelCubo>
                  </NivelesCubo>
                   ...
            </DimensionCubo>
      </DimensionesCubo>

      <Medidas>
            <Medida>
               ...
            </Medida>
      </Medidas>

      <Roles>
            <Rol>
              ...
            </Rol>
      </Roles>

      <Particiones>
            <Particion>
                  <Agregaciones>
                        <Agregacion>
                              <DimensionesAgregacion>
                                    <DimensionAgregacion>
```

```
                                        <NivelesAgregacion>
                                              <NivelAgregacion>
                                                    ...
                                              </NivelAgregacion>
                                        </NivelesAgregacion>
                                            ...
                                    </DimensionAgregacion>
                                </DimensionesAgregacion>
                               ...
                           </Agregacion>
                      </Agregaciones>
                  ...
            </Particion>
      </Particiones>

      <NombresCubosBase>
              ...
      </NombresCubosBase>
</CuboActual>
```

5.2.7 Módulo Importador

Este componente es el encargado de reconstruir cubos, previamente exportados, en el Servidor OLAP que se indique. Para llevar a cabo este objetivo, expone un único método denominado *importar*, que ejecuta esta tarea en forma completamente transparente. Ya que toda la información requerida para efectuar este proceso se encuentra en el objeto *Registro* que recibe como parámetro, se logra una total independencia en su implementación.

Se pueden dividir los cubos a importar en tres clases disjuntas:

1. Cubos regulares
2. Cubos virtuales, exportados previamente sin sus cubos base
3. Cubos virtuales, exportados previamente con sus cubos base

Internamente el método que expone este módulo, se divide en tres subrutinas especializadas en un cada uno de estos casos particulares; dependiendo de la naturaleza del cubo a importar se invoca alguna de ellas.

Cada una de estas rutinas al comenzar, presentan interfaces al usuario que le permiten escoger el servidor OLAP destino, así como el nombre de base de datos en donde será generado el cubo. El propósito fundamental de estas interfaces, es establecer la conexión con el servidor destino y proporcionar a la subrutina de importación correspondiente, un nombre validado de base de datos OLAP a utilizar. Esto permite liberar al subproceso de importación de algunos detalles de validación, facilitando su algoritmo.

Antes de importar la metadata de un cubo regular, es necesario replicar el almacén de datos ligado al mismo. Por tal motivo, se despliega una nueva interface gráfica que permite al usuario tomar esta decisión, así como indicar el destino de la replicación.

La información referente al almacén de datos vinculado a un cubo regular, está especificada en su *cadena de conexión a datos*. Esta cadena (contenida en el objeto *OrigenDeDatos* de nuestro modelo de cubo), consiste en una lista de tokens separados por el carácter ";" que detalla todos los parámetros de conexión utilizados por el cubo. Procesando cuidadosamente esta cadena, se puede obtener el nombre del almacén de datos, el servidor SQL en el cual se encuentra, así como el nombre de usuario y contraseña que permiten el acceso al almacén. Los datos así obtenidos, en conjunto con el destino de replicación indicado por el usuario, son proporcionados a los métodos adecuados de la clase *CargadorDeDatos* que se encargan de construir el paquete de transformación de datos *(DTS)* y ejecutar el proceso de migración entre los servidores SQL de manera transparente.

Las cadenas de conexión de cada cubo, son posteriormente actualizadas, para que referencien al nuevo almacén de datos.

Es interesante señalar que la subrutina que implementa la importación de un cubo virtual y sus cubos vinculados merece un tratamiento especial, puesto que no solo consiste en importar la estructura multidimensional (que se mantiene en memoria) al servidor OLAP destino, sino que se debe hacer lo propio con cada cubo base.

Básicamente, para tratar este caso, se procede a cargar y posteriormente importar cada cubo base en forma iterativa. Puesto que el objeto *Registro* contiene todas las rutas absolutas a las instancias XML en donde están almacenados estos cubos vinculados, se utilizan los servicios que provee el componente *CargadorDeMetadataExp* para generarlos de a uno, con el propósito de importar su metadata y replicar su almacén de datos. Una vez finalizada la iteración, se procede a reconstruir el cubo virtual en el destino.

Ya que normalmente, algunos cubos vinculados a un cubo virtual pueden estar basados en el mismo almacén de datos, se toman las medidas pertinentes para mantener la consistencia entre los orígenes de datos de los cubos.

Como se puede apreciar, es necesario interactuar con todos los componentes relevantes de la aplicación *(CargadorDeMetadataOS, CargadorDeDatos, CargadorDeMetadataExp)* para implementar el proceso de importación, que culmina con el procesamiento (optativo) del cubo.

CAPÍTULO 6

Conclusiones

6. Conclusiones

6.1 Resumen de lo hecho

A la vista de los resultados finales, creemos que la aplicación producto de este proyecto cumple con los objetivos pautados.

Los principales logros obtenidos se enumeran a continuación:

- **Migración de metadata de estructuras multidimensionales**

  La estructura de un cubo puede ser obtenida desde un servidor Microsoft OLAP Services y posteriormente recreada en otro servidor OLAP del mismo tipo. Como paso intermedio en este proceso, la metadata es llevada a un formato XML.

- **Migración de datos**

  La base de datos relacional vinculada a un cubo, también puede ser migrada desde el servidor origen hacia el servidor destino especificado.
  Como se ha hecho referencia en capítulos previos, esta funcionalidad obliga a que ambos servidores sean Microsoft SQL Server y estén accesibles desde la aplicación durante el proceso de importación.

- **Estructura del cubo expresada en XML**

  La herramienta trabaja sobre un formato propio expresado en XML. Las instancias generadas por el producto son válidas de acuerdo al esquema suministrado en cada proceso de exportación. Este esquema fue desarrollado con el propósito de proveer la especificación en la que se basan los archivos XML generados por el prototipo.

- **Visualización de la estructura de un cubo en un browser**

  Un cubo expresado en formato XML, puede ser posteriormente visualizado en un browser estándar. Para lograr este cometido, se desarrollaron dos hojas de estilo cuyo objetivo es transformar la información contenida en las instancias XML correspondientes en archivos HTML. Estos archivos son finalmente integrados en una página web especialmente diseñada para este fin.

- **Modificaciones a los distintos elementos del cubo**

La aplicación soporta un subconjunto de las operaciones admitidas por OLAP Services, permitiendo efectuar también ciertos cambios no implementados por el propio servidor OLAP. Estas modificaciones al modelo de cubo pueden ser luego reflejadas en su representación en XML.

- **Aciertos en el diseño**

Procuramos que el diseño fuera lo suficientemente flexible y simple, de manera de facilitar el mantenimiento de la aplicación así como permitir la incorporación de nuevas funcionalidades. En consecuencia se definió una arquitectura modular, intentando agrupar en componentes independientes las rutinas de interacción con las distintas tecnologías que fueron utilizadas.

- **Programa autosuficiente**

El programa puede ser instalado en equipos que no posean un servidor OLAP o algún software particular. En estos casos, las modificaciones a la metadata de un cubo se efectúan en el equipo local, accediendo únicamente a servidores remotos durante el proceso de migración.

6.2 Temas pendientes

A pesar de lo citado existen ciertas limitaciones en la herramienta obtenida, ya que quedaron a un lado una serie de elementos que hubiera sido deseable incluir.

En el caso de futuros trabajos sobre lo desarrollado, entre los temas que sería interesante abordar se encuentran los siguientes:

- migración de bases de datos entre servidores que no sean necesariamente Microsoft SQL Server. Esto es particularmente deseable ya que los cubos en Microsoft OLAP Services permiten un conjunto variado de orígenes de datos y nosotros nos estamos limitando a migrar los datos de cubos que se encuentran en servidores Microsoft SQL Server.

- modificación de las distintas propiedades que poseen los elementos que componen la metadata del cubo, de las cuales actualmente solo se está permitiendo modificar un pequeño subconjunto.

- permitir desde la interfaz gráfica agregar nuevos elementos a un cubo que ha sido exportado, cuya estructura se encuentra expresada en XML. Estos nuevos elementos podrían ser dimensiones, medidas, particiones, miembros calculados, entre otros.

- como se hizo alusión en otros puntos del informe, la base de datos en la que se basa un cubo es directamente transferida desde un servidor SQL Server a otro, sin pasar por un estado intermedio (por el que sí pasa la metadata del cubo). Este estado intermedio podría ser incorporado, en el cual la base de datos estaría expresada en un archivo XML. A través de esta funcionalidad se le otorgaría a los datos del cubo la misma portabilidad adjudicada a su metadata al utilizar la nueva representación.

- actualmente el programa es capaz de llevar un cubo que ha sido exportado, a un formato apto para ser visualizado en un browser. Esta idea podría extenderse de modo que también los datos del cubo pudieran ser llevados a un formato similar. Esto permitiría poder apreciar en un browser tanto la metadata de un cubo como sus datos.

- incorporar en el programa una utilidad que permita examinar los datos del cubo además de su metadata. Con este avance se podrían especificar rebanadas del cubo y así poder observar sus datos, permitiendo juzgar la consistencia de los mismos.

- permitir diseñar agregados sobre cubos que han sido exportados, esto se podría realizar (tal como lo hace Microsoft OLAP Services) permitiendo al usuario especificar requerimientos en cuanto a performance y espacio de almacenamiento.

- dar la posibilidad de crear cubos directamente en el nuevo formato, involucrando esta operación el acceso a bases de datos y la creación de los distintos elementos que constituyen la estructura de un cubo.

Bibliografía

Libros

- Microsoft OLAP Solutions
  Autores: Erik Thomsen
  George Spofford
  Dick Chase

- SQL Server 7, Data Warehousing
  Autores: Michael Corey
  Michael Abbey
  Ian Abramson
  Larry Barnes
  Benjamin Taub
  Rajan Venkitachalam

Manuales

- Manual de XML
  Autores: Charles F. Goldfarb
  Paul Prescod

- Programación de SQL Server 7.0 con Visual Basic 6.0

- Microsoft Visual Basic 6.0, Manual de programador

Libros en línea y artículos

- Microsoft SQL Server 7.0

- Microsoft OLAP Services

- Microsoft XML Parser SDK

- Microsoft Developer Network Library

- Document Object Model (Core) Level 1 Recomendation (octubre 1998)

- Microsoft Windows DNA XML Resource Kit

APÉNDICE

OLAP Services

OLAP Services

1. Introducción al sistema

Los servicios OLAP de Microsoft SQL Server proporcionan una arquitectura de proceso analítico en línea (OLAP, *Online Analytical Processing*) que permite el acceso rápido a los datos contenidos en un almacén de información. Los datos almacenados en el almacén de información se extraen, resumen y almacenan en estructuras multidimensionales para dar rápida respuesta a las consultas de los usuarios.

Los servicios OLAP y PivotTable proporcionan la capacidad para diseñar, crear y administrar cubos contenidos en almacenes de datos y permiten el acceso de clientes a los datos de OLAP. El servidor OLAP administra los datos; el servicio PivotTable trabaja con el servidor para proporcionar a los clientes acceso a los datos.

1.1 Arquitectura del cubo OLAP

El objeto OLAP básico es el cubo, que consiste en una representación multidimensional de datos de detalle y resumen. Un cubo consta de un origen de datos, dimensiones, medidas y particiones. Los cubos se diseñan a partir de los requisitos de análisis de los usuarios. Un almacén de datos puede contener muchos cubos distintos, por ejemplo, el cubo Ventas, el cubo Inventario, etc.

El origen de datos del cubo identifica y se conecta con la base de datos donde se encuentra el almacén de datos, que es el origen de los datos del cubo.

Las dimensiones asignan la información de la tabla de dimensiones del almacén de datos a una jerarquía de niveles, por ejemplo, una dimensión Geográfica con los niveles Continente, País, Estado o provincia, y Ciudad. Las dimensiones se pueden crear de forma independiente y se pueden compartir entre los cubos para facilitar la construcción de éstos y asegurar la consistencia del resumen de datos de análisis. Por ejemplo, si se utiliza una dimensión compartida para una jerarquía de productos en los cubos apropiados, la organización de la información resumida de los productos será coherente entre los distintos cubos que utilicen esa dimensión.

Una dimensión virtual es un tipo especial de dimensión que asigna las propiedades de los miembros de otra dimensión a una dimensión que, a partir de ese momento, se puede utilizar en cubos. Por ejemplo, una dimensión virtual de la propiedad tamaño de un

producto permite que un cubo resuma datos como, por ejemplo, la cantidad de ventas por producto y por tamaño. Las dimensiones virtuales y las propiedades de los miembros se evalúan como corresponda para ejecutar las consultas y no necesitan un espacio físico de almacenamiento en el cubo.

Las medidas identifican los valores numéricos extraídos de la tabla de hechos que están resumidas para realizar el análisis.

Las particiones son los contenedores multidimensionales de almacenamiento que guardan los datos del cubo. Cada cubo contiene, al menos, una partición y los datos de un cubo se pueden combinar a partir de varias particiones. Cada partición puede obtener sus datos de un origen de datos distinto y se pueden almacenar en una ubicación diferente. Los datos de una partición se pueden actualizar independientemente de los datos contenidos en otras particiones del mismo cubo. Por ejemplo, los datos de un cubo se pueden dividir mediante criterios de tiempo, con una partición para los datos del año actual, otra partición para los datos del año anterior y una tercera partición para los datos más antiguos.

Las particiones de un cubo se pueden almacenar separadas en distintos modos de almacenamiento con diferentes grados de resumen. Las particiones son invisibles para el usuario, para quien el cubo aparece como un único objeto, pero siguen proporcionando al administrador una amplia variedad de opciones para administrar los datos de OLAP subyacentes.

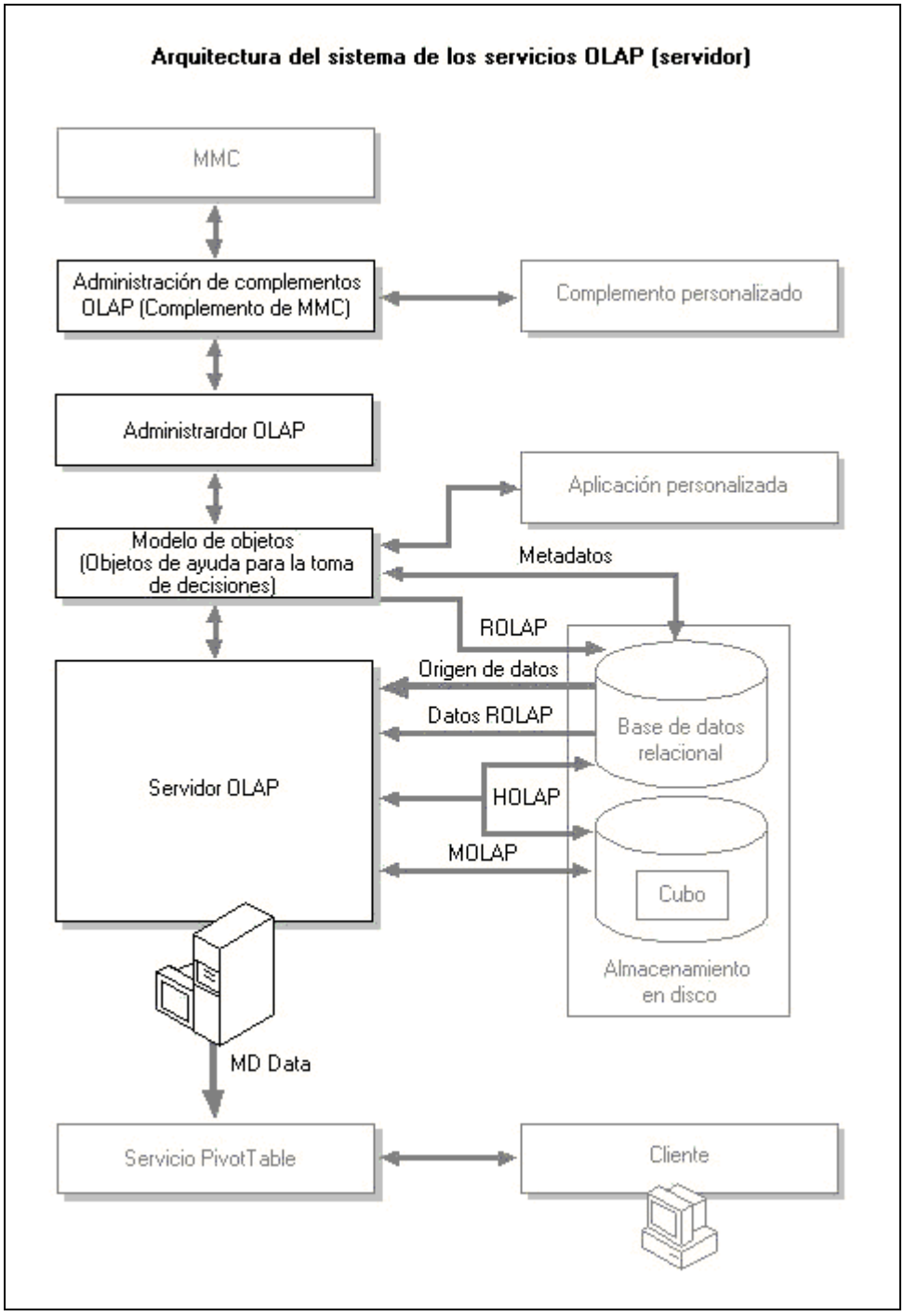
Las funciones permiten administrar el acceso de los usuarios a los datos del cubo al asignar las cuentas de grupos y de usuarios de Microsoft Windows NT a privilegios de acceso al cubo.

Un cubo virtual es una vista lógica de porciones de uno o más cubos. Un cubo virtual puede utilizarse para combinar cubos relativamente distintos que comparten una dimensión común, por ejemplo un cubo Ventas y un cubo Almacén, con propósitos de análisis, al mismo tiempo que se conserva la funcionalidad propia de cada cubo. Se pueden seleccionar dimensiones y medidas de los cubos combinados para presentarlas en el cubo virtual.

1.2 Arquitectura del servidor

Los servicios OLAP de Microsoft SQL Server proporcionan capacidades de servidor para crear y administrar datos de OLAP multidimensionales y para suministrar datos a clientes mediante el servicio PivotTable. Entre las operaciones del servidor se incluyen la creación de cubos de datos multidimensionales a partir de bases de datos y almacenes de datos relacionales, y el almacenamiento de cubos en estructuras multidimensionales de cubos, en bases de datos relacionales o en una combinación de ambas. Los metadatos de las estructuras multidimensionales de cubos se almacenan en un depósito de una base de datos relacional.

El complemento Administrador OLAP proporciona una interfaz de usuario que se ejecuta como un complemento en Microsoft Management Console (MMC). Las interfaces de programación se proporcionan para permitir que las aplicaciones personalizadas interactúen con el modelo de objetos que controla el servidor y con el Administrador OLAP.

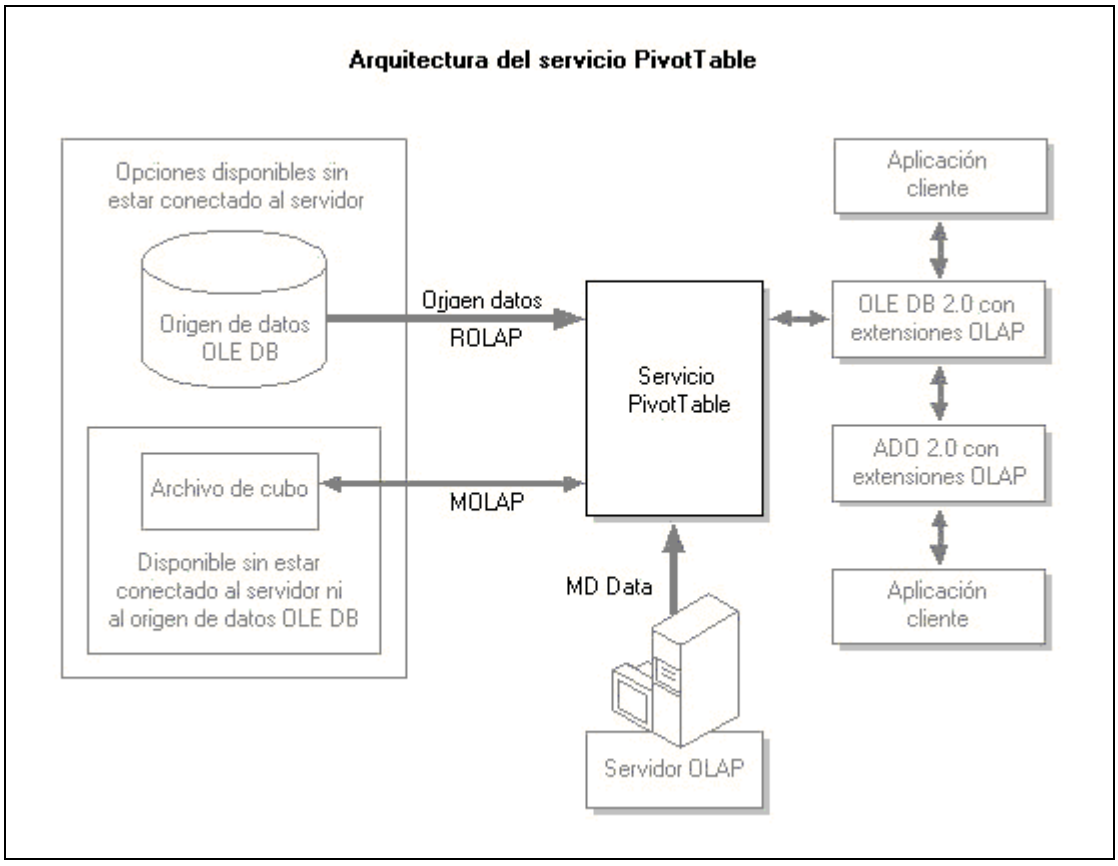


1.3 Arquitectura del cliente

El servicio PivotTable se comunica con el servidor OLAP y proporciona interfaces que las aplicaciones de cliente utilizan para tener acceso a los datos OLAP almacenados en el servidor.

Las aplicaciones de cliente se conectan con el servicio PivotTable mediante interfaces OLE DB para C++ o mediante el modelo de objetos de Microsoft ActiveX Data Objects (ADO) para lenguajes de automatización del Modelo de objetos componentes (COM, *Component Object Model*), como Microsoft Visual Basic.

Los servicios PivotTable también pueden crear archivos de cubos locales que contienen datos de un cubo del servidor o de las bases de datos relacionales OLE DB. Los cubos locales se pueden almacenar como archivos de cubos multidimensionales en el equipo cliente. Los cubos locales se pueden utilizar sin conexión gracias al servicio PivotTable para análisis portátiles.



2. Acerca de los almacenes de datos

En muchas ocasiones, un almacén de datos se utiliza como el fundamento de un sistema de ayuda para la toma de decisiones. Se ha diseñado para superar algunos de los problemas que una organización encuentra cuando intenta realizar un análisis estratégico mediante la misma base de datos que utiliza para realizar el proceso de transacciones en línea (OLTP). Normalmente, los sistemas OLTP:

- Admiten el acceso simultáneo de muchos usuarios que agregan y modifican datos.
- Representan el estado constantemente cambiante de una organización, pero no guardan su historial.
- Contienen grandes cantidades de datos, incluidos los datos extensivos utilizados para comprobar transacciones.
- Tienen estructuras complejas.
- Se ajustan para dar respuesta a la actividad transaccional.
- Proporcionan la infraestructura tecnológica necesaria para admitir las operaciones diarias de la empresa.

A continuación se enumeran algunas de las dificultades que se suelen encontrar al utilizar bases de datos OLTP para realizar análisis en línea:

- Los analistas carecen de la experiencia técnica necesaria para crear consultas "ad hoc" contra la compleja estructura de datos.
- Las consultas analíticas que resumen grandes volúmenes de datos afectan negativamente a la capacidad del sistema para responder a las transacciones en línea.
- El rendimiento del sistema cuando está respondiendo a consultas analíticas complejas puede ser lento o impredecible, lo que causa un servicio poco eficiente a los usuarios del proceso analítico en línea.
- Los datos que se modifican con frecuencia interfieren en la coherencia de la información analítica.
- La seguridad se hace más complicada cuando se combina el análisis en línea con el proceso de transacciones en línea.

El almacén de datos proporciona una de las claves para resolver estos problemas al organizar los datos para el propósito del análisis. Los almacenes de datos:

- Pueden combinar datos de orígenes heterogéneos en una única estructura homogénea.
- Organizan los datos en estructuras simplificadas buscando la eficiencia de las consultas analíticas más que del proceso de transacciones.
- Contienen datos transformados que son válidos, coherentes, consolidados y con el formato adecuado para realizar el análisis.
- Proporcionan datos estables que representan el historial de la empresa.

- Se actualizan periódicamente con datos adicionales, no con transacciones frecuentes.
- Simplifican los requisitos de seguridad.
- Proporcionan una base de datos organizada para OLAP, no para OLTP.

Un puesto de datos es un formato especial de almacén de datos que suele contener un subconjunto de datos de la empresa orientado a un tema determinado y apropiado para una función empresarial específica.

La versión 7.0 de Microsoft SQL Server proporciona muchas herramientas, incluidos los Servicios de transformación de datos (DTS, *Data Transformation Services*), esenciales para construir almacenes de datos y puestos de datos.

3. Acerca de OLAP

Mientras que los almacenes y puestos de datos son los almacenes donde se analizan los datos, el Proceso analítico en línea (OLAP) es la tecnología que permite a las aplicaciones de cliente el acceso eficiente a estos datos. OLAP proporciona muchas ventajas a los usuarios que realizan análisis, por ejemplo:

- Un modelo de datos intuitivo y multidimensional que facilita la selección, recorrido y exploración de los datos.
- Un lenguaje analítico de consulta proporciona la capacidad de explorar las complejas relaciones existentes entre los datos empresariales.
- El precálculo de los datos consultados con más frecuencia permite una rápida respuesta a las consultas ad hoc.
- Una eficaz herramienta ayuda a los usuarios a crear nuevas vistas de datos basadas en una rica matriz de funciones de cálculo ad hoc.

La versión 7.0 de los Servicios OLAP de Microsoft SQL Server es una robusta herramienta OLAP que se puede utilizar con los datos almacenados en diversas bases de datos, incluidas las de SQL Server, Microsoft Access y Oracle.

4. OLAP y los almacenes de datos

OLAP proporciona una presentación multidimensional de los datos de un almacén mediante la creación de cubos que organizan y resumen los datos para mejorar la eficiencia de las consultas analíticas. El diseño de la estructura del almacén de datos puede afectar a la facilidad con la que se podrán diseñar y construir estos cubos.

La versión 7.0 de los Servicios OLAP se basa en que los datos proporcionados por el almacén son seguros, estables y tienen integridad referencial.

4.1 Cubos

Los cubos son elementos clave en el proceso analítico en línea (OLAP, *Online Analytic Processing*). Los cubos proporcionan un mecanismo de consulta de datos con un tiempo de respuesta corto y uniforme sin importar la cantidad de datos almacenados en el cubo o la complejidad de la consulta.

Los cubos son subconjuntos de datos del almacén de datos, organizados y resumidos en estructuras multidimensionales. Los resúmenes de datos, precalculados según factores empresariales seleccionados, proporcionan el mecanismo para obtener tiempos de respuesta cortos y uniformes para las consultas complejas.

4.2 Medidas

Las medidas son datos numéricos de gran interés para los usuarios de los cubos. Las medidas que se seleccione dependerán de los tipos de información que solicitan los usuarios. Algunas medidas comunes son ventas, costos, gastos, cuenta de producción, etc.
Los Servicios OLAP de Microsoft SQL Server agregan las medidas y los agregados se almacenan para que los usuarios que consultan los cubos puedan recuperarlos rápidamente.

Cada medida se almacena en una columna de la tabla de hechos en el almacén de datos. Debido a que un cubo puede contener únicamente una tabla de hechos, todas las medidas del cubo deben estar contenidas en ella.

Una medida puede contener varias columnas combinadas en una expresión. Por ejemplo, la medida Beneficios es la resta de dos columnas numéricas: Ventas y Costos.

Las columnas de la tabla de hechos pueden ser aditivas o no aditivas, y los Servicios OLAP pueden utilizar ambos tipos como medidas. Las columnas *aditivas* se pueden sumar. Por ejemplo, una columna monetaria es aditiva. Las columnas aditivas son apropiadas para utilizarse como medidas de un cubo. La suma de columnas *no aditivas* carece de significado. Por ejemplo, una columna numérica que contiene un identificador como *Número de cuenta* no es aditiva. Las columnas no aditivas no son apropiadas como medidas de un cubo, pero se pueden combinar con ayuda de ciertas funciones como *Count*. El resultado se puede utilizar después como una medida.

Para aplicar una función a una columna no aditiva de forma que el resultado se pueda utilizar como una medida, se debe crear un miembro calculado.

Los miembros calculados se pueden utilizar como medidas. Los valores de los miembros calculados se crean mediante fórmulas cuando se examina el contenido del cubo, pero estos

valores no se almacenan. De esta forma, los miembros calculados ahorran espacio de almacenamiento en disco.

4.3 Almacenar cubos y agregados

Se pueden almacenar los datos y agregados de un cubo en una amplia variedad de formas. Los agregados son resúmenes precalculados de datos que constituyen un mecanismo apropiado para obtener respuestas rápidas a consultas en los sistemas OLAP.

Los cubos pueden necesitar una cantidad considerable de espacio de almacenamiento para contener los datos y la información de resumen precalculada en estructuras multidimensionales.

Un factor que afecta a los requisitos de almacenamiento es la dispersión (la cantidad de celdas vacías existentes en un cubo). Los Servicios OLAP satisfacen los requisitos de almacenamiento en el cubo mediante una amplia diversidad de técnicas:

- Las celdas vacías no tienen espacio de almacenamiento asignado.
- Se emplea la compresión de datos.
- Un sofisticado algoritmo diseña eficientes agregados de resúmenes para minimizar el espacio de almacenamiento sin sacrificar la velocidad.

Las opciones de almacenamiento permiten seleccionar modos y ubicaciones apropiados para almacenar los datos del cubo.

4.3.1 Modos de almacenamiento (MOLAP, ROLAP, HOLAP)

Las opciones de almacenamiento físico afectan al rendimiento y a los requisitos de almacenamiento de los cubos. Se puede almacenar un cubo en una estructura MOLAP (OLAP multidimensional), en una base de datos ROLAP (OLAP relacional), o en una combinación HOLAP (OLAP híbrido) de estructura multidimensional y base de datos relacional. Cada estrategia de almacenamiento presenta ventajas y desventajas.

El almacenamiento MOLAP utiliza una estructura multidimensional para contener agregados y una copia de los datos base. El sistema de almacenamiento MOLAP proporciona los tiempos de respuesta a consultas más rápidos, que dependen únicamente del porcentaje y del diseño de los agregados del cubo. En general, MOLAP es más apropiado para cubos de uso frecuente y que necesitan tiempos de respuesta muy cortos.

El almacenamiento ROLAP utiliza tablas en las bases de datos relacionales del almacén de datos para almacenar los agregados de un cubo. A diferencia del almacenamiento MOLAP, ROLAP no almacena una copia de los datos base, sino que tiene acceso a la tabla de hechos original cuando es necesario para responder a consultas. Las respuestas a consultas ROLAP suelen ser más lentas que aquellas que se realizan con otras dos estrategias de almacenamiento.

Un uso típico de ROLAP es el acceso a grandes conjuntos de datos consultados con poca frecuencia, tales como datos históricos de años no recientes.

El sistema de almacenamiento HOLAP combina atributos de MOLAP y ROLAP. Los datos de los agregados se almacenan en estructuras MOLAP mientras que los datos base se mantienen en la base de datos relacional del almacén de datos. Para las consultas que tienen acceso a los datos de resumen, HOLAP es el equivalente de MOLAP. Las consultas que tienen acceso a datos base, por ejemplo una consulta que aumenta el nivel de detalle hasta un hecho simple, deben recuperar los datos de la base de datos relacional y no se ejecutarán con tanta rapidez como cuando los datos base están almacenados en la estructura MOLAP. Los cubos almacenados como HOLAP tienen un tamaño menor que los cubos MOLAP equivalentes y responden con mayor rapidez que los cubos ROLAP a consultas relativas a datos de resumen. El almacenamiento HOLAP suele ser adecuado para cubos que requieren tiempos cortos de respuesta para consultas realizadas en resúmenes basados en grandes cantidades de datos base.

Los Servicios OLAP admiten los tres modos de almacenamiento. El Asistente para almacenamiento y agregado de datos proporciona opciones que permiten seleccionar el modo de almacenamiento más adecuado para el cubo.

4.4 Particiones

Los cubos se pueden dividir en particiones, cada una de las cuales se puede almacenar mediante un modo distinto (MOLAP, ROLAP, HOLAP). Las particiones de un cubo son invisibles para el usuario, sin embargo, es importante que las particiones se definan de tal manera que contengan datos mutuamente exclusivos. Un cubo puede proporcionar resultados incorrectos a algunas consultas si una parte de los datos del cubo está incluida en más de una partición.

Las particiones de un cubo se pueden almacenar en varios servidores para proporcionar un método de almacenamiento en cubos basado en clústeres.

Dos particiones de un cubo pueden mezclarse en una única partición que, a su vez, puede combinarse con otra partición y así sucesivamente hasta que quede una única partición. Por ejemplo, se puede mezclar cuatro particiones, cada una de las cuales contienen datos correspondientes a un trimestre, en una única partición que contenga los datos de todo el año.

4.5 Agregados

Los agregados son resúmenes de datos precalculados que mejoran el tiempo de respuesta a las consultas por el simple hecho de tener preparadas las respuestas antes de que se planteen las preguntas. Por ejemplo, la respuesta a una consulta que solicita el total de ventas semanales de una determinada línea de productos y que se realiza en una tabla de hechos de un almacén de datos que contiene cientos de miles de filas de información, puede llevar mucho tiempo si hay que explorar la tabla de hechos para calcular la respuesta. Por el contrario, la respuesta podría ser casi inmediata si los datos de resumen para la respuesta a esta consulta se han calculado previamente. El cálculo previo de los datos de resumen es la clave para obtener respuestas rápidas en la tecnología OLAP. Si se calculan previamente todos los posibles agregados a un cubo, se obtiene el tiempo de respuesta más corto posible para todas las consultas. Sin embargo, el tiempo de almacenamiento y el tiempo de proceso necesarios para todos los agregados puede ser sustancial.

Los requisitos de almacenamiento no sólo dependen del número de dimensiones y medidas, sino también del número de niveles de las dimensiones y del número de miembros de cada nivel.

Existe un equilibrio entre los requisitos de almacenamiento y el porcentaje de posibles agregados que se han precalculado. Si no se calcula previamente ningún agregado (0%), se necesitará poco espacio de almacenamiento además del necesario para almacenar los datos base. Sin embargo, en este caso, los tiempos de respuesta a consultas pueden variar y pueden llegar a ser muy elevados, ya que será necesario calcular todas las respuestas a partir de los datos de base en cada consulta.

Los Servicios OLAP incorporan un sofisticado algoritmo que permite la selección de agregados para realizar cálculos previos, de forma que se puedan calcular rápidamente otros agregados a partir de los valores ya calculados. Por ejemplo, si se han calculado previamente agregados para el nivel mes de una dimensión temporal, los cálculos para un trimestre implicarán tan sólo el resumen de tres números, operación que se puede realizar con gran rapidez. Esta técnica ahorra espacio de almacenamiento y afecta muy poco al tiempo de respuesta de las consultas.

4.6 Procesar cubos

Cuando se procesa un cubo, se calculan los agregados diseñados para el cubo y se carga el cubo con los agregados y datos calculados. Procesar un cubo implica leer las tablas de dimensiones para llenar los niveles con miembros de los datos actuales, leer la tabla de hechos, calcular los agregados específicos y almacenar los resultados en el cubo. Una vez que se ha procesado un cubo, pueden consultarlo los usuarios.

Según las circunstancias, puede utilizar una de las tres siguientes opciones de proceso:

- Procesar
- Actualización incremental
- Actualizar los datos

Procesar un cubo

Procesar es el término que se utiliza para una carga completa del cubo. Se leen todos los datos de las dimensiones y de la tabla de hechos, y se calculan los agregados especificados. Se debe procesar un cubo cuando su estructura sea nueva o cuando se hayan modificado sus dimensiones o medidas. El proceso de un cubo puede llevar mucho tiempo si existe una tabla de hechos de grandes dimensiones y hay muchas dimensiones con muchos niveles y muchos elementos en cada nivel.

Los cambios en el esquema del almacén de datos que afectan a la estructura de los cubos requieren que la estructura de dichos cubos se cambie y, después, se procese. Los cambios o los datos agregados al almacén de datos no requieren que los cubos se procesen completamente.

Estos cambios se pueden incorporar a los cubos existentes mediante las opciones de procesamiento **Actualización incremental** o **Actualizar los datos**, en función de la forma en que hayan cambiado los datos.

Actualizar incrementalmente un cubo

Una actualización incremental es adecuada cuando se van a agregar nuevos datos al cubo pero los datos existentes no cambian y la estructura del cubo sigue siendo la misma. La opción **Actualización incremental** agrega nuevos datos y actualiza agregados; no requiere un proceso completo del cubo.

Una actualización incremental no afecta a los datos existentes que ya se han procesado. Requiere bastante menos tiempo que la ejecución de un procesamiento. Una actualización incremental se puede llevar a cabo mientras los usuarios consultan el cubo; una vez finalizada la actualización, los usuarios tienen acceso a los datos adicionales sin necesidad de desconectarse y volverse a conectar.

- **Actualizar los datos de un cubo**

La opción **Actualizar los datos** hace que se borren y vuelvan a cargar los datos de un cubo, y se vuelvan a calcular los agregados. Esta opción es adecuada cuando se han modificado los datos subyacentes contenidos en el almacén de datos pero se ha conservado la estructura del cubo. Esta opción es más rápida que procesar el cubo porque no es necesario volver a diseñar las tablas de agregados.

4.7 Dimensiones

Las *dimensiones* son categorías descriptivas que se utilizan para facilitar el análisis de los datos numéricos (es decir, las medidas) contenidos en un cubo. Por ejemplo, si una medida de un cubo es Cuenta de producción y sus dimensiones son Fecha, Factoría, Ubicación y Producto, los usuarios del cubo pueden dividir Cuenta de producción en varias categorías de Fecha, Factoría, Ubicación y Producto.

Se puede crear una dimensión para utilizarla en un sólo cubo o en varios cubos. Una dimensión creada para un cubo individual se denomina *dimensión privada.* En la vista de árbol del Administrador OLAP, las dimensiones privadas se encuentran en la carpeta Dimensiones del cubo para el que se crearon. Una dimensión que varios cubos pueden utilizar recibe el nombre de *dimensión compartida*. Se almacenan en la carpeta **Dimensiones compartidas** de la carpeta **Biblioteca** de la base de datos. Las dimensiones compartidas también aparecen en las carpetas Dimensiones de los cubos que las contienen.

Las dimensiones compartidas pueden utilizarse en cualquier cubo de la base de datos. Al crear dimensiones compartidas y utilizarlas en varios cubos se ahorra tiempo ya que no se necesita crear las mismas dimensiones privadas en cada uno de los cubos. Las dimensiones compartidas también permiten la estandarización de medidas empresariales utilizadas entre cubos. Por ejemplo, las dimensiones compartidas estandarizadas para fechas y ubicación geográfica aseguran que los datos analizados desde cubos distintos tendrán una organización similar.

También se puede crear otro tipo de dimensión conocida como dimensión virtual, que se detallará posteriormente.

Cuando se defina una dimensión, se debe seleccionar una o más columnas de una tabla de dimensiones. Si selecciona varias columnas, todas ellas deben estar relacionadas entre sí de tal forma que sus valores se puedan organizar en una única jerarquía.

Cada columna de una dimensión constituye un *nivel* en la dimensión. Los niveles se ordenan por nivel de detalle y se organizarán en una jerarquía que permita la creación de caminos lógicos para el aumento del nivel de detalle. Cada nivel contiene miembros. Los *miembros* son los valores de la columna que define el nivel.

Cada clave principal de una tabla de dimensiones debe combinarse con una clave externa de la tabla de hechos de un cubo o de otra tabla de dimensiones. Para definir una dimensión no se necesitan las columnas de clave.

4.8 Dimensiones virtuales

Una dimensión virtual es una dimensión lógica basada en una propiedad de un nivel correspondiente a una dimensión física. Esta propiedad se denomina "propiedad de miembro" y debe crearse antes que la dimensión virtual.

Las dimensiones virtuales permiten que los usuarios finales analicen los datos del cubo basados en las propiedades de miembros del nivel de la dimensión de un cubo. Los usuarios finales podrán utilizar esta dimensión como cualquier otra.

Al agregar la dimensión virtual al cubo no aumenta el tamaño del mismo porque, a diferencia de lo que ocurriría en una dimensión normal, los datos de agregado de una dimensión virtual no se almacenan en el cubo. Las dimensiones virtuales no afectan al tiempo de proceso del cubo porque se calculan en memoria cuando se necesitan. Sin embargo, las consultas que utilicen las dimensiones virtuales pueden ser significativamente más lentas que las consultas que utilicen dimensiones normales.

4.9 Miembros calculados

Un *miembro calculado* es un miembro de dimensión cuyo valor se calcula en tiempo de ejecución mediante una expresión que el usuario especifica al definir el miembro calculado. Los miembros calculados también se pueden definir como medidas.

Sólo se almacenan las definiciones de los miembros calculados; los valores se calculan en memoria cuando se necesite dar respuesta a una consulta.

Los miembros calculados le permitirán agregar miembros y medidas a un cubo sin aumentar su tamaño. Aunque los miembros calculados deben estar basados en los datos existentes de un cubo (tales como miembros), se pueden crear expresiones complejas si se combina estos datos con operadores aritméticos, números y una amplia variedad de funciones proporcionadas por los Servicios OLAP.

4.10 Cubos virtuales

Un cubo virtual es una combinación de varios cubos en un único cubo lógico, algo similar a lo que ocurre con una vista de una base de datos relacional que combina otras vistas y tablas.

Para crear un cubo virtual, se deben seleccionar las medidas y dimensiones del conjunto consolidado de dimensiones y medidas de los cubos que van a intervenir en el proceso (cubos base). Los usuarios finales verán al cubo virtual como un único cubo.

Un cubo virtual puede estar basado en un único cubo para exponer únicamente los subconjuntos seleccionados de sus medidas y dimensiones.

Debido a que los cubos virtuales almacenan únicamente sus definiciones y no los datos de sus cubos componentes, apenas requieren espacio físico de almacenamiento. El propósito de estos cubos virtuales es, entonces, crear combinaciones y variantes de los cubos existentes sin utilizar espacio adicional de almacenamiento.

Un cubo virtual puede mejorar la seguridad del sistema sin más que limitar la vista de los cubos subyacentes a ciertos usuarios.

4.11 Propiedades de miembros

Una propiedad de miembro es un atributo de un miembro de dimensión. Proporciona a los usuarios del cubo información adicional acerca del miembro.

Las propiedades de miembros tienen diversos usos. Además de proporcionar información acerca de un miembro, las propiedades de miembros pueden utilizarse en consultas para proporcionar así a los usuarios más opciones cuando analizan los datos del cubo. Las propiedades de miembros son también la base de las dimensiones virtuales.

Los valores de las propiedades de miembros se deben leer de una columna contenida en la misma tabla de dimensiones que los miembros asociados.

4.12 Cubos habilitados para escritura

Si especifica un cubo como habilitado para escritura, las aplicaciones de cliente podrán registrar los cambios a los datos del cubo. Estos cambios reciben el nombre de datos de "preescritura" y se almacenan en una tabla separada del cubo y de sus datos, pero se incorporarán en los resultados de las consultas como si fueran parte de los datos del cubo. Los cubos habilitados para escritura permiten a los usuarios finales explorar escenarios con sólo cambiar los valores de las celdas y analizar los efectos que han tenido los cambios en los datos del cubo.

Cualquier cambio realizado por un usuario final se almacena en una tabla de preescritura a diferencia de lo que ocurre con los valores mostrados. Por ejemplo, si un usuario final cambia el valor de una celda (de 90 a 100), en la tabla de preescritura se almacena el valor +10 junto con la hora del cambio e información acerca del usuario final que lo llevó a cabo. En las aplicaciones de cliente se mostrará el efecto neto de los cambios acumulados. Se conservará el valor original del cubo y en la tabla de preescritura se registrará un recorrido de auditoría de los cambios.

Sólo se podrán efectuar cambios en celdas del nivel inferior, es decir, las celdas que no contengan información de agregados. Por ejemplo, un usuario final puede cambiar el valor de una venta individual pero no el total de las ventas correspondiente a un almacén, trimestre o cualquier otro valor de resumen. Los cambios se aplican a los valores resumidos cuando se evalúan las consultas para que los usuarios finales puedan ver los efectos del cambio en todo el cubo.

Los cambios realizados por los usuarios finales se almacenan en una tabla de preescritura independiente que se podrá:

- Convertir en una partición para incorporar permanentemente los cambios en el cubo y convertirlo en un cubo de sólo lectura.
- Descartar los cambios para devolver el cubo a su estado original.

Cualquier usuario final podrá registrar los cambios en una tabla de preescritura de un cubo sólo si pertenece a una función que tenga los permisos de lectura y escritura asignados al cubo.

APÉNDICE B

Lenguaje Extensible de Marcas (XML)

Lenguaje Extensible de Marcas (XML)

1. Introducción

XML, lenguaje extensible de etiquetas (extensible por que no es un formato prefijado como HTML), describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas que los procesan. XML por tanto es ante todo un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos. Los documentos XML se componen de unidades de almacenamiento llamadas entidades (*entities*), que contienen datos analizados (*parsed*) o sin analizar (*unparsed*). Los datos analizados se componen de caracteres, algunos de los cuales forman los datos del documento y el resto forman las etiquetas. Las etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento. XML proporciona un mecanismo para imponer restricciones en la estructura lógica y de almacenamiento.

Cualquier aplicación que trabaje con XML necesita un módulo software llamado procesador XML. Su función es leer documentos y proporcionar acceso a su contenido y estructura. Para poder llevar a cabo esta función, la aplicación debe proporcionar información al procesador XML de cómo se encuentra almacenada esta información a través de un DTD.

2. Objetivos

Los objetivos de diseño de XML son:

- XML debería ser directamente utilizable en Internet.
- XML debería soportar una amplia variedad de aplicaciones.
- XML debería ser compatible con SGML.
- Debería ser sencillo escribir programas que procesaran documentos XML.
- El número de características opcionales de XML debería ser el mínimo, a ser posible cero.

- Los documentos XML deberían ser legibles para las personas y razonablemente claros.
- El diseño del XML debería finalizar de forma rápida.
- XML debería ser simple pero perfectamente formalizado.
- Los documentos XML deberían ser sencillos de crear.
- La brevedad de las marcas XML es de mínima importancia.

3. Documentos XML

Un objeto de datos es un documento XML si esta bien formado. Un documento XML bien formado puede además ser válido si cumple una serie de restricciones.

Todo documento XML posee una estructura lógica y una física. Físicamente, el documento está compuesto de unidades llamadas entidades. Una entidad puede hacer referencia a otras entidades para que éstas sean incluidas en el documento. Un documento comienza en una "raíz" o entidad documento. Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos los cuales se indican en el documento mediante marcas explícitas. Las estructuras lógica y física deben anidarse de manera correcta.

Los documentos XML se dividen en dos grupos, documentos bien formados y documentos válidos.

- **Bien formados:** son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas, sin estar sujetos a unos elementos fijados en un DTD (más adelante se explicará que es un DTD). De hecho los documentos XML deben tener una estructura jerárquica muy estricta, y los documentos bien formados deben cumplirla.
- **Válidos:** Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD: sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

3.1 Estructuras lógicas

Cada documento XML contiene uno o más elementos, cuyos límites están delimitados por etiquetas de comienzo y de final o, en el caso de elementos vacíos, por una etiqueta de elemento vacío.

Cada elemento tiene un tipo, identificado por un nombre, denominado identificador genérico, y puede tener un conjunto de especificaciones de atributos.

Cada especificación de atributo tiene un nombre y un valor. Estas especificaciones no restringen la semántica, el uso o (mas allá de la sintaxis) los nombres de los tipos de los elementos y los atributos, excepto de los nombres que comienzan por XML, que se reservan para estandarizar etiquetas o atributos en versiones posteriores del estándar.

3.2 Estructuras físicas

Un documento XML puede consistir en una o más unidades de almacenamiento virtual, llamadas entidades. Todas estas unidades tienen contenido y todas ellas (excepto la entidad documento y el subconjunto externo del DTD) están identificadas por un nombre. Cada documento XML contiene una entidad, llamada entidad documento, que sirve como punto de comienzo para el procesador XML y puede contener el documento completo.

Las entidades pueden ser analizadas o sin analizar (también llamadas procesadas o sin procesar). El contenido de una entidad analizada se conoce también como texto de reemplazo, y es parte integrante del documento. Las entidades no analizadas son recursos cuyo contenido puede o no ser texto, o en caso de que sea texto que no sea XML. Cada entidad no analizada tiene una notación asociada, identificada por un nombre. Aparte de obligar al procesador XML a que haga accesible a la aplicación el nombre de esta notación y sus identificadores asociados, XML no proporciona ninguna otra restricción sobre el contenido de estas entidades. La forma de invocar ambos tipos de entidades es a través de su nombre, en el caso de las analizadas a través de su referencia a entidad y en el de las no analizadas a través de sus atributos de entidad.

Las entidades generales, son entidades analizadas que se usan en el interior del documento. Las entidades parametrizadas son entidades analizadas que se usan en el ámbito del DTD. Estos dos tipos de entidades usan distintos tipos de referencias y se reconocen en contextos distintos.

4. Declaración de Tipo de Documento (DTD)

El DTD o declaración del tipo de documento *(Document Type Declaration)* proporciona la gramática para una clase de documentos XML. Esta gramática contiene la definición del conjunto de etiquetas que puede contener esa clase de documentos XML. Un DTD es generalmente un fichero (o varios usados conjuntamente) que contiene la definición formal de un tipo de documento particular. Éste define los nombres que pueden utilizarse en los elementos, dónde pueden aparecer y cómo se interrelacionan entre ellos.

5. Esquemas XML

Un esquema define los elementos que pueden aparecer dentro de un documento y los atributos que pueden estar asociados a dichos elementos. También define la estructura de un documento: cuales elementos son elementos hijos de otros, la secuencia en que los elementos hijos pueden aparecer, y el número de éstos elementos hijos. Define cuando un elemento está vacío o puede incluir texto. Un esquema puede también definir los valores por defecto de los atributos.

Los esquemas XML, como DTD, pueden ser usados para especificar un esquema de una clase específica de documentos. Sin embargo, a diferencia de DTD, los esquemas XML usan sintaxis XML. Esto es muy conveniente ya que implica que no sea necesario tener que aprender una sintaxis completamente nueva para poder usar esquemas XML (aunque se necesite aprender como declarar elementos y atributos usando esquemas XML).

Algunas ventajas de la utilización de esquemas XML son las siguientes:

- Admiten tipos de datos enriquecidos, lo que permite una validación más ajustada de los datos y una reducción del esfuerzo a la hora de aplicarlos.
- A través de la utilidad de espacios de nombre, los esquemas XML mejoran la expresividad, garantizando la existencia de nombres aceptados de forma única.
- Ofrecen utilidades semánticas mayores y más eficaces al incorporar el concepto de herencia, permitiendo así que un esquema se base en otro.

Puesto que los esquemas XML utilizan la sintaxis de XML, se benefician de una serie de ventajas:

- Las mismas herramientas que se utilizan para analizar XML pueden utilizarse para analizar la notación de los esquemas XML.

- Ya que la sintaxis es muy parecida a HTML, los autores de HTML no tendrán problemas para entenderla y leerla.

- Puede extenderse de forma más sencilla.

El parser XML (a partir de la versión 2.0) que proporciona Internet Explorer 5 puede validar documentos XML tanto con DTD como con esquemas XML.

6. Hojas de estilo (XSL)

SGML tiene su propio estándar para la representación de sus documentos, el DSSSL *(Document Style Semantics and Specification Language)*, que en realidad es un lenguaje de programación completo y muy potente basado en el dialecto Scheme del lenguaje LISP. Por tanto, ya que XML es una versión reducida de SGML parecía lógico hacer también una versión reducida del DSSSL, llamada en este caso XSL *(Extended Stylesheet Language)*.

Básicamente, XSL es un lenguaje de hojas de estilos diseñado para su utilización en WWW. Provee una funcionalidad más allá de las hojas de estilo en cascada (CSS o Cascade Stylesheet) del HTML dinámico. De momento no existe una versión definitiva del mismo (de hecho, ni siquiera se sabe con seguridad si como lenguaje base utilizará LISP o Java), aunque si se conoce la funcionalidad mínima que debe proporcionar: representar de forma independiente a la plataforma utilizada o al medio de representación la información recogida en los documentos XML. Dicha representación es creada mediante la formación de un árbol de objetos de flujo *(flow objects tree)*. Un objeto de flujo tiene una clase, la cual representa una tarea o actividad de representación. Asimismo, un objeto de flujo dispone de un conjunto de características, mediante las cuales se puede especificar mucho más la representación. La asociación de elementos en el árbol origen con los objetos de flujo se lleva a cabo mediante las reglas de construcción, las cuales contienen un patrón para identificar elementos específicos en el árbol origen, y una acción para especificar un subárbol resultante de objetos de flujo.

El procesador de hojas de estilo procesará de forma recursiva los elementos del origen para producir un completo árbol de objetos de flujo. Además de las reglas de construcción, XSL también soporta reglas de estilo, las cuales permiten la mezcla de características. Mientras que una sola regla de construcción puede ser invocada para un elemento en particular del origen, todas las reglas de estilo aplicables son invocadas, permitiendo la mezcla de características como en CSS.

Resumiendo, una hoja de estilo XSL describe el proceso de presentación a través de un pequeño conjunto de elementos XML. Esta hoja, puede contener elementos de reglas que representan a las reglas de construcción y elementos de reglas de estilo que representan a las reglas de mezcla de estilos.

Los formatos gráficos a soportar son problema de los visualizadores específicamente, y por tanto de la definición del XSL específico. XML por sí mismo ni predice ni restringe nada. Los formatos que se perfilan como de mayor utilidad son GIF, JPG, TIFF y CGM.

7. Herramientas para trabajar con XML

7.1 Editores XML

Un editor XML es una herramienta que ofrece facilidades para crear y editar documentos XML. Inicialmente podemos trabajar con un simple editor de textos si estamos familiarizados con la sintaxis y características del XML aunque es recomendable un editor específico. Dentro de estos, existen dos tipos principales:

- Los que representan los ficheros en forma de árbol y nos permiten construir nuestro documento trabajando sobre este árbol y formularios adicionales (por ejemplo XML Notepad de Microsoft o Visual XML).
- Los que representan el documento XML en su formato original y que normalmente son editores de ficheros de texto con facilidades para XML (por ejemplo XED o PSGML de Emacs).

Entre ambos tipos hay que diferenciar los que trabajan con una DTD y por lo tanto validan el contenido de lo que escribimos y los que simplemente nos aseguran que el documento XML es bien formado, es decir, sintácticamente correcto respecto de las especificaciones del XML.

7.2 Editores de DTD y editores de esquemas XML

Como se ha comentado anteriormente, en una DTD se define como es la estructura de un documento XML, es decir, los elementos que formarán ese tipo de documento y como están relacionados. A diferencia de SGML, en XML no es obligatorio crearla, aunque es recomendable porque nos facilitará la validación de documentos. Algunos ejemplos de estos editores son el TDTD para Emacs o el EZDTD que también trabaja con SGML.

Así como existen editores de DTD también los hay de esquemas XML. Como ejemplo de estas herramientas se puede citar a XML Authority.

7.3 Procesadores XML

El parser o procesador de XML es la herramienta principal de cualquier aplicación XML. Mediante este parser no solamente podemos comprobar si nuestros documentos son bien formados o válidos, sino que también podemos incorporarlos a nuestras aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML.

De acuerdo con su función, los procesadores XML se dividen en dos tipos: validadores y no-validadores. En común tienen que ambos deben informar de las violaciones de las restricciones de documento bien formado dadas en su especificación. Además, los procesadores validadores, deben informar de la violación de las restricciones expresadas por las declaraciones del DTD.

Ya están disponibles varios de estos parsers XML como el LT XML de la Universidad de Edimburgo, el MSXML de Microsoft o el TclXML de Steve Ball.

8. Aplicaciones y ventajas

Algunas de las aplicaciones de XML son:

- Ofrecer mecanismos más versátiles de mostrar datos. Actualmente, bajo el nombre de DOM *(Document Object Model)* se está desarrollando una API que sea soportada por todos los procesadores de XML y HTML. La idea detrás de esta API es que podamos representar (a través de JavaScripts o JavaApplets) documentos XML en los navegadores web, pero de una forma más sofisticada que los documentos HTML, ya que XML no solo proporciona una sintaxis, sino también una semántica.

- Buscadores inteligentes. Debido a que la información en los documentos XML está etiquetada por su significado de forma precisa, podemos localizarla de forma mucho más clara que en documentos HTML. Con DTD's estandarizados para distintas aplicaciones (librerías, tiendas de deporte, catálogos de componentes, etc.) podríamos programar buscadores web que recuperasen información sobre un producto de cualquier website en el mundo sabiendo que todos tendrán el mismo formato de datos (gracias al DTD), aunque no tengan necesariamente la misma representación gráfica (gracias al XML/XSL).

- Intercambio de información entre sistemas heterogéneos. El fundamento es el mismo que para los buscadores inteligentes. Debido a que el DTD proporciona un formato estándar para representar la información de un tema específico, puede usarse para simplificar el intercambio de información entre distintas fuentes (actualmente existen ya dos DTD estandarizados, uno para fabricantes de chips y otro para industrias químicas, llamado CML).

Algunas de las ventajas del XML:

- Los autores y proveedores pueden diseñar sus propios tipos de documentos usando XML, en vez de limitarse a HTML. Los tipos de documentos pueden ser explícitamente “hechos a la medida de una audiencia”, por lo que las difíciles manipulaciones que se deben hacer con HTML para conseguir efectos especiales serán cosa del pasado: autores y diseñadores serán libres de inventar sus propias etiquetas.
- La información contenida puede ser más “rica” y fácil de usar, porque las habilidades hipertextuales de XML son mayores que las de HTML.
- XML puede dar más y mejores facilidades para la representación en los visualizadores.
- Elimina muchas de las complejidades de SGML, en favor de la flexibilidad del modelo, con lo que la escritura de programas para manejar XML será más sencilla que haciendo el mismo trabajo en SGML.
- La información será más accesible y reutilizable, porque la flexibilidad de las etiquetas de XML pueden utilizarse sin tener que amoldarse a reglas específicas de un fabricante, como es el caso de HTML.
- Los ficheros XML válidos son válidos también en SGML, luego pueden utilizarse también fuera de la Web, en un entorno SGML (una vez la especificación sea estable y el software SGML la adopte).

APÉNDICE C

Manual de usuario

Manual de usuario

1. Introducción

El presente manual pretende describir las distintas funcionalidades de *Migrador OLAP* y la forma en que estas se llevan a cabo. Para tal fin, en las siguientes secciones se ilustrará la interfaz de la aplicación, su contenido, así como las facilidades que esta ofrece para la realización de las distintas operaciones.

Primeramente se hará mención de las funcionalidades que la aplicación provee, para luego pasar a detenerse en cada una de estas para dejar en claro la forma en que pueden ser realizadas.

2. Funcionalidades provistas por Migrador OLAP

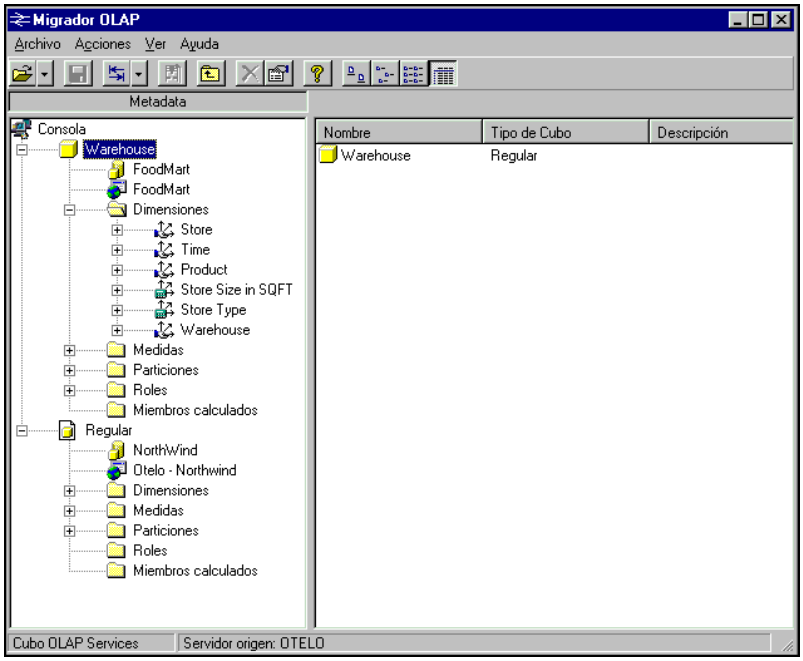
Las funcionalidades que brinda la aplicación se listan a continuación:

- **Visualizar la metadata de un cubo en la interfaz de la aplicación**
- **Exportar un cubo**
- **Importar un cubo**
- **Generar un archivo para ser visualizado en un browser**
- **Eliminar un elemento de un cubo**
- **Renombrar un elemento de un cubo**
- **Cambiar la descripción de un elemento de un cubo**
- **Especificar si una dimensión es compartida o privada**
- **Especificar si una medida es oculta o no**
- **Especificar el modo de almacenamiento de una partición**

3. Utilización de Migrador OLAP

Desde la ventana principal proporcionada por *Migrador OLAP* es posible observar la metadata de un determinado cubo, realizar las modificaciones admitidas sobre sus elementos, llevar a cabo exportaciones e importaciones, así como el resto de las funcionalidades que proporciona la aplicación.

Ventana principal de Migrador OLAP



La ventana principal contiene los siguientes elementos: barra de tareas, barra de estado, panel de la izquierda y panel de la derecha. Estos elementos serán descritos en las secciones que siguen.

3.1 Barra de tareas

Los botones en la barra de tareas junto con las funciones a las cuales permiten acceder se listan a continuación:

- **Abrir cubo (1)**
  Despliega en la interfaz de la aplicación la metadata de un cubo que se encuentra en un servidor Microsoft OLAP Services, o bien un cubo que ha sido exportado previamente.

- **Guardar (2)**
  En el caso de que se hayan realizado cambios en el cubo seleccionado, al accionar este botón se guardan dichos cambios en el cubo exportado correspondiente. Es de mencionar que también se brinda una funcionalidad que permite guardar el cubo en una ubicación distinta a la que este se encuentra, esta operación se realiza desde el menú archivo con la opción *Guardar como...* .

- **Exportar-Importar (3)**
  Efectúa la exportación o importación, según corresponda, del cubo seleccionado. Se realizará una exportación si se trata de un cubo que se encuentra en un servidor OLAP, y una importación si se trata de un cubo que ha sido exportado.

- **Generar un archivo para visualización (4)**
  Crea un archivo que posibilita visualizar la metadata del cubo seleccionado en un browser.

- **Subir un nivel (5)**
  En la estructura jerárquica que expresa la metadata de un cubo en la interfaz de la aplicación, deja como seleccionado el elemento de nivel inmediatamente superior al elemento que está seleccionado actualmente.

- **Eliminar elemento (6)**
  Elimina el elemento seleccionado de la metadata de un cubo (en caso de que la eliminación de dicho elemento esté soportada por la aplicación).

- **Propiedades (7)**
  Despliega un formulario en el que se exponen distintas propiedades del elemento de la metadata seleccionado.

- **Temas de ayuda (8)**
  Muestra la ayuda de la aplicación.

- **Vista de iconos (9)**

  Determina el modo en que serán visualizados los iconos en el panel de la derecha. Los modos posibles son: iconos grandes, iconos pequeños, lista y detalles.

Barra de tareas

3.2 Barra de estado

Aquí se indica si el cubo seleccionado ha sido exportado o no. Esto se expone a través de la frase *Cubo OLAP Services*, en el caso de que se trate de un cubo que no se ha exportado, y *Cubo Exportado* en caso contrario. También se muestra cual es el origen del cubo seleccionado desplegando el nombre del servidor OLAP en el que reside, si se trata de un cubo no exportado, o la ruta al lugar en disco en donde se encuentra, si es un cubo que ya ha sido exportado.

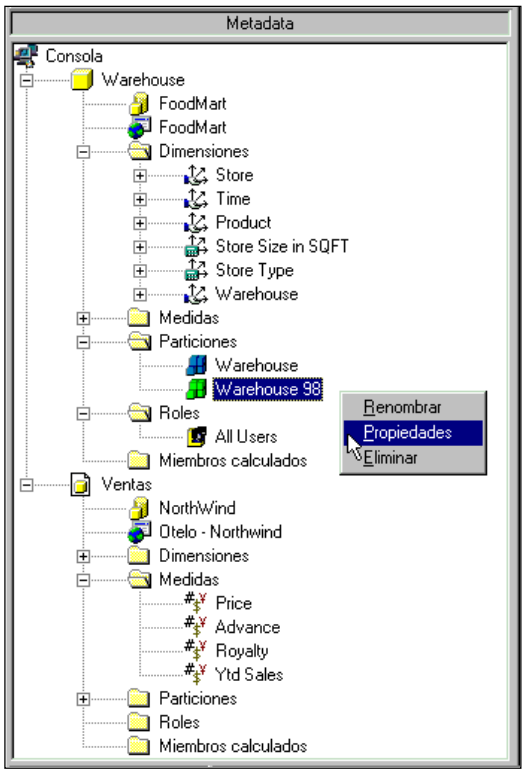
Barra de estado

3.3 Panel de la izquierda y panel de la derecha

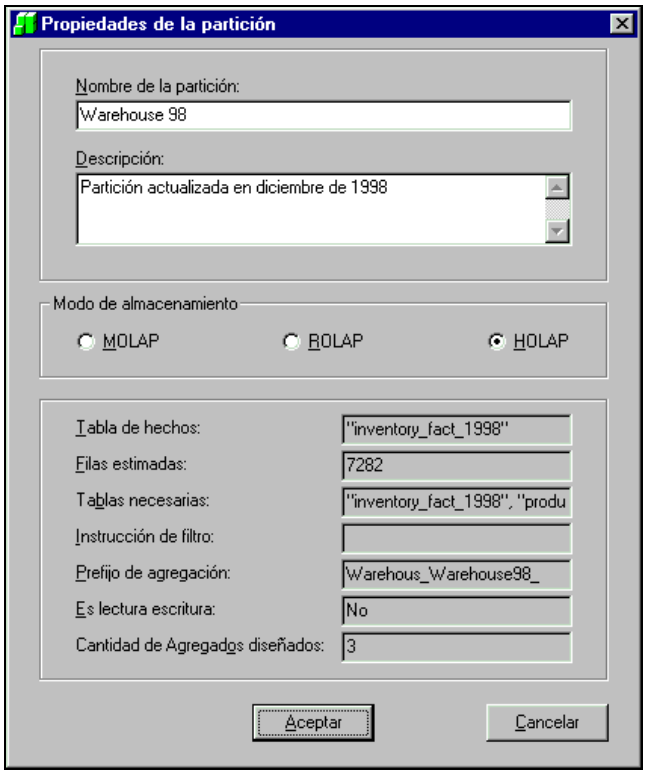
En estos paneles se puede visualizar la metadata de los cubos que se encuentran actualmente abiertos en la aplicación. Los distintos elementos se encuentran organizados en estructuras jerárquicas, desde las cuales es posible acceder a los mismos para realizar sobre ellos las diversas operaciones disponibles.

Luego de seleccionado el elemento, las operaciones se pueden realizar utilizando la barra de tareas, el menú de la aplicación o bien el menú contextual. Entre las opciones disponibles se encuentra la que permite apreciar las propiedades del elemento seleccionado. Al activar esta opción se despliega un formulario que contiene algunas de las propiedades del elemento.

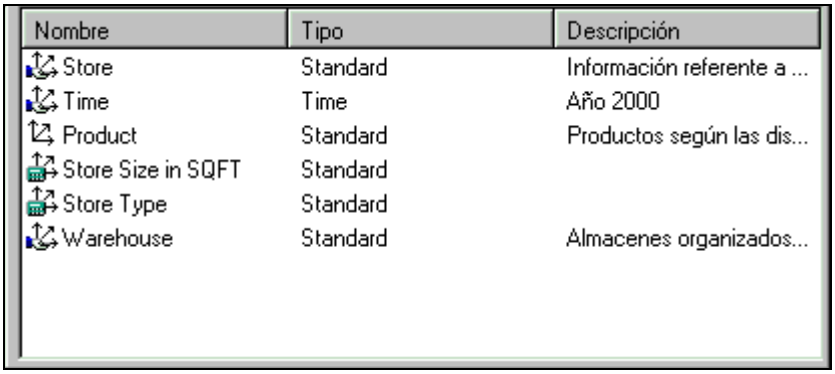
Panel izquierdo



Formulario de propiedades de una partición



Panel derecho



Nombre	Tipo	Descripción
Store	Standard	Información referente a ...
Time	Time	Año 2000
Product	Standard	Productos según las dis...
Store Size in SQFT	Standard	
Store Type	Standard	
Warehouse	Standard	Almacenes organizados...

4. Visualizar la metadata de un cubo en la interfaz de la aplicación

Esta operación es la que permite desplegar la metadata de un cubo en la interfaz de la aplicación. Los cubos a ser visualizados en la interfaz pueden ser cubos que residan en servidores Microsoft OLAP Services, o cubos que ya han sido exportados.

En el caso de que se trate de un cubo que todavía no ha sido exportado, los pasos para realizar la operación son los siguientes:

1. Accionar el botón *Abrir cubo* de la barra de herramientas (o bien la opción *Abrir Cubo OLAP Services...* del menú *Archivo*). Esta acción hace que se abra una ventana que permite seleccionar el servidor en el cual reside el cubo deseado.

Cuadro de conexión con un servidor OLAP

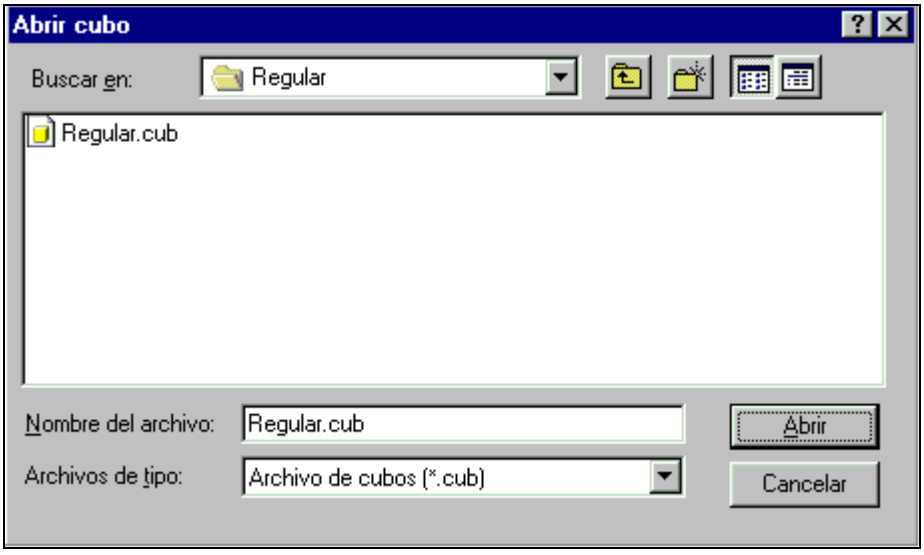
2. Luego de presionar el botón *Aceptar*, se despliega una nueva ventana en la que se pueden apreciar todos los cubos que residen en el servidor especificado, organizados de acuerdo a las bases de datos a las que pertenecen. Aquí se selecciona un cubo y al presionar el botón *Aceptar*, la metadata del mismo se despliega en la interfaz de la aplicación.

Cuadro de selección de un cubo desde un servidor OLAP

Para cubos que han sido exportados, los pasos a seguir son los siguientes:

1. Accionar el botón *Abrir cubo* de la barra de herramientas (o bien la opción *Abrir Cubo Exportado...* del menú *Archivo*). Esta acción hace que se abra una ventana que permite seleccionar el archivo correspondiente al cubo deseado.

Cuadro para abrir un cubo que ha sido exportado



2. Luego de seleccionar el archivo y presionar el botón *Aceptar* de la ventana previa, la metadata del cubo se despliega en la interfaz de la aplicación.

5. Exportar un cubo

Esta operación permite exportar un cubo que reside en un determinado servidor OLAP, a un nuevo formato especificado en XML. Brindando la posibilidad de especificar el lugar de residencia del cubo exportado.

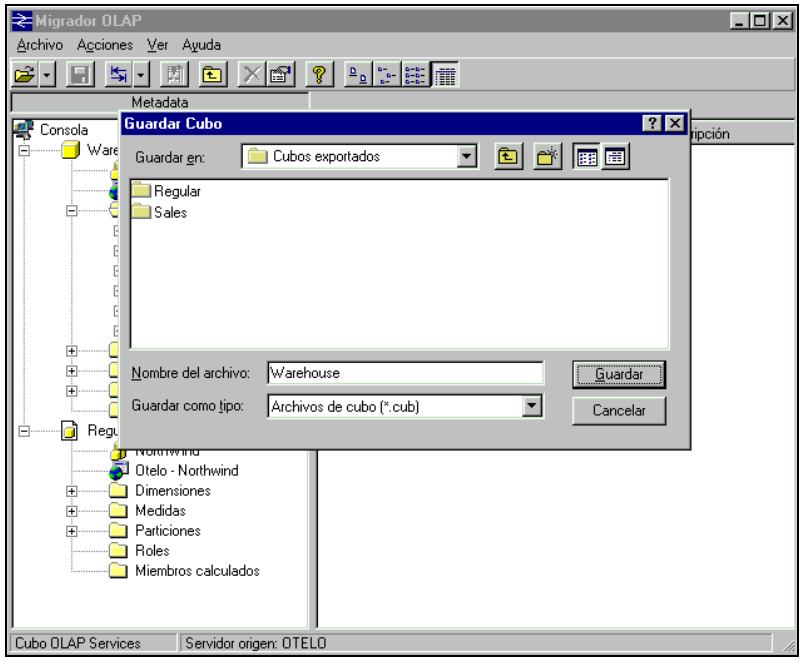
Si el cubo que se está exportando es virtual, la operación también permite manifestar si desea o no exportar los cubos base asociados.

Llevar a cabo la operación de exportación involucra los siguientes pasos:

1. Seleccionar un cubo de la interfaz de la aplicación que todavía no ha sido exportado.

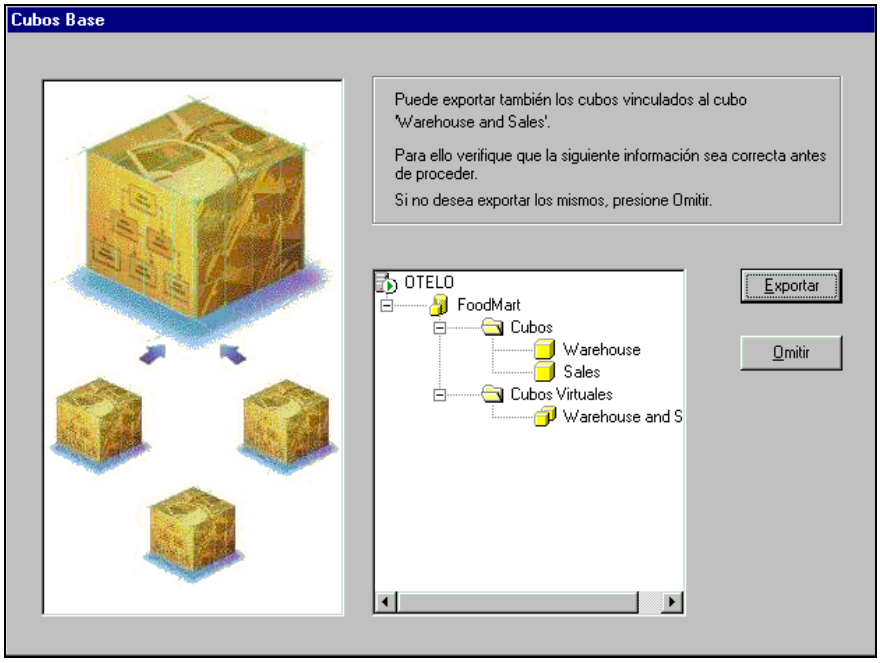
2. Accionar el botón *Exportar-Importar* de la barra de tareas (o bien seleccionar la opción *Exportar* del menú *Acciones*). Al hacer esto se abrirá un cuadro de diálogo donde podrá especificar en donde desea guardar el cubo al exportarlo.

Cuadro de selección del lugar de residencia del cubo a exportar



3. Si el cubo a exportar es un cubo virtual, se despliega un formulario desde donde se puede especificar si se desea o no exportar también los cubos base del cubo virtual en cuestión.

Cuadro de exportación de un cubo virtual



Se debe seleccionar *Omitir* si se quiere que los cubos base no sean exportados y *Exportar* en caso contrario.

Finalmente, la exportación se comienza a llevar a cabo y luego de concluida se notifica si fue concretada con éxito.

Al exportar un cubo se generan en la ubicación especificada los siguientes archivos:

- ***nombre_de_cubo.xml***

  Contiene todos los datos acerca de la metadata del cubo *nombre_de_cubo.* En caso de cubos virtuales se genera un archivo .xml por cada uno de sus cubos base, en la situación de que éstos se hayan exportado conjuntamente con su virtual.

- ***Cubo.xdr***

  Contiene el esquema al que se ajustan las instancias de los archivos .xml generados.

- ***Cubo.xsl***
  Hoja de estilo utilizada para visualizar la estructura del esquema *Cubo.xdr* en un browser.

- ***nombre_seleccionado.cub***
  Siendo *nombre_seleccionado* el nombre dado a la exportación.
  Este archivo contiene información adicional acerca del cubo exportado.

6. Importar un cubo

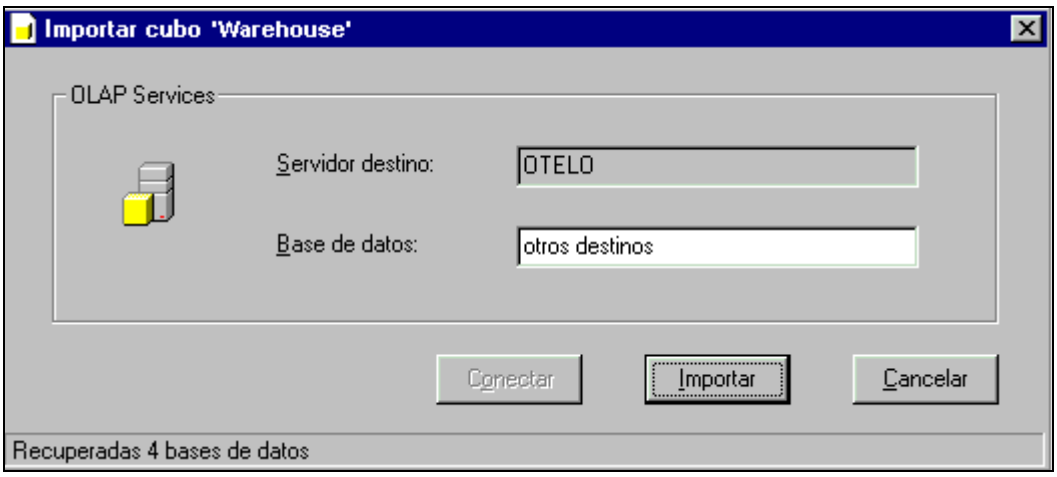
Esta operación consiste en importar un cubo expresado en el formato XML a un servidor especificado, obteniendo un cubo en la representación utilizada por Microsoft OLAP Services. Si el cubo es regular, en el momento de realizar la importación se puede expresar si se quiere importar la base de datos en la que está basado el cubo en cuestión o si solo se importará su metadata. Si se trata de un cubo virtual, se tiene la posibilidad de realizar además la importación de los cubos base correspondientes, siempre que estos se hayan incluido en el proceso de exportación. Para los distintos cubos base, se podrá seleccionar si es deseado importar sus bases de datos.

La importación de un cubo involucra los siguientes pasos:

1. Seleccionar de la interfaz de la aplicación un cubo que ha sido exportado.

2. Accionar el botón *Exportar-Importar* de la barra de tareas (o bien seleccionar la opción *Importar* del menú *Acciones*). Al realizar esta operación se abrirá un cuadro de diálogo donde podrá especificar el servidor y la base de datos destino donde será reconstruido el cubo.

Cuadro de selección del destino de la importación de un cubo

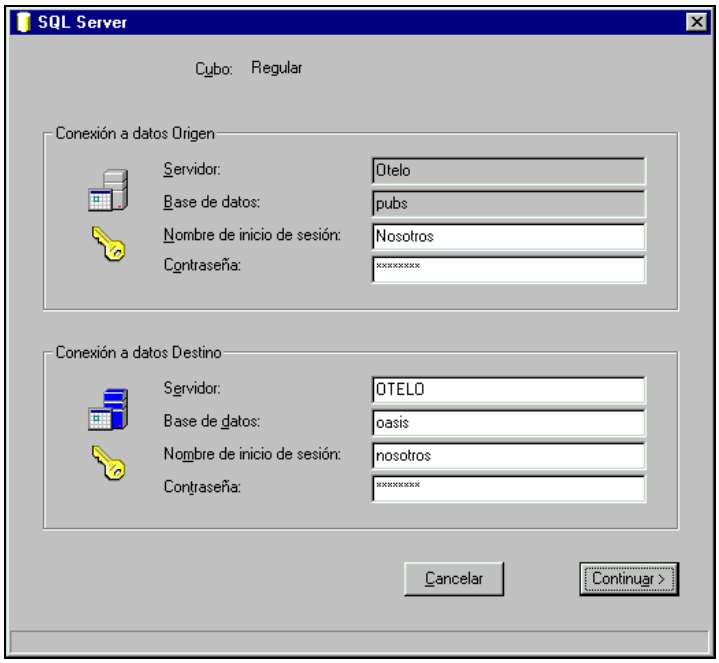


Para cubos regulares, en el caso de que se seleccione una base de datos que no exista en el servidor OLAP, la misma será creada para luego generar el nuevo cubo en ella. Si se selecciona una base de datos existente, la operación no se podrá concretar en el caso de que en dicha base de datos exista un cubo con el mismo nombre que el que se quiera agregar. Si la base de datos existente contiene dimensiones u orígenes de datos que coinciden en nombre con algunas dimensiones u orígenes de datos a agregar, estos elementos no serán creados, tomándose en su lugar los ya existentes (si dichos elementos no son iguales a los que poseía el cubo originalmente se puede generar un cubo que no trabaje correctamente). Las dimensiones existentes que tienen orígenes de datos diferentes al del cubo que se está generando no serán utilizadas en dicho cubo.

Para los cubos virtuales, si éstos se importan sin sus cubos base, en la base de datos destino deben estar disponibles todas las medidas y dimensiones necesarias. Si la importación se realiza conjuntamente con los cubos base, siempre se debe crear una base de datos nueva en el servidor destino.

3. En el caso de que en el proceso de importación se haya generado un nuevo origen de datos, luego de presionar el botón *Importar* en el cuadro comentado en el punto anterior, se despliega un formulario que permite especificar si se desea replicar la base de datos en la que se basa el cubo que se está importando. Para realizar tal operación se debe especificar: el servidor SQL Server destino, el nombre de la base de datos destino (en el servidor destino no debe existir una base de datos con este nombre), y el nombre de usuario y contraseña necesarios para acceder al servidor especificado.

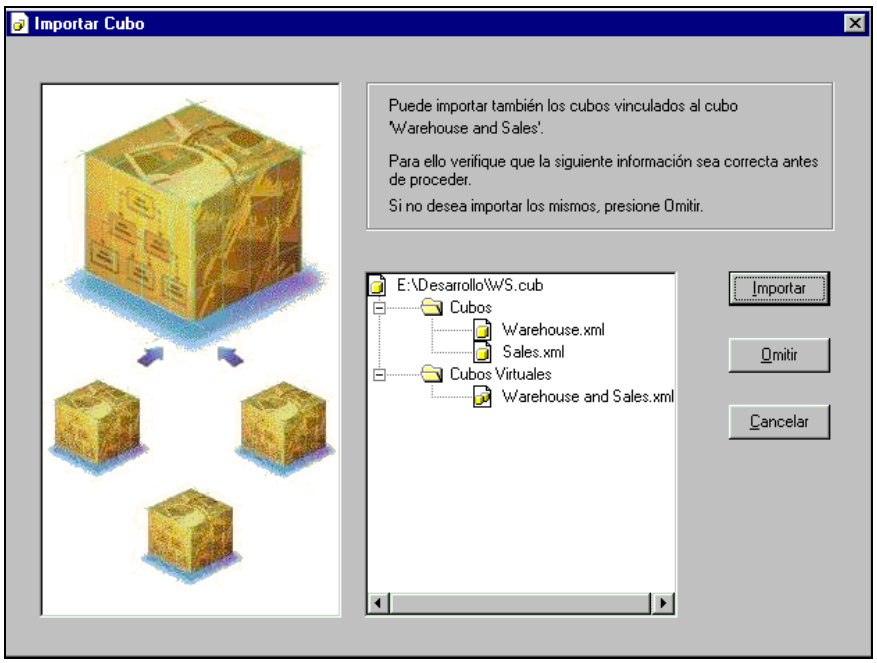
Cuadro de migración de la base de datos en la que se base el cubo a importar



Es de destacar que el formulario anterior solo se proporciona si el origen de datos del cubo referencia a una base de datos que se encuentra en un servidor Microsoft SQL Server, y solo podrá ser migrada hacia un servidor del mismo tipo.

Si se está trabajando sobre un cubo virtual, y este se exportó conjuntamente con sus cubos base, inmediatamente luego de iniciar la operación de importación se proporciona un formulario desde el cual se podrá especificar si desea importar dichos cubos base.

Cuadro de importación de un cubo virtual conjuntamente con sus cubos base

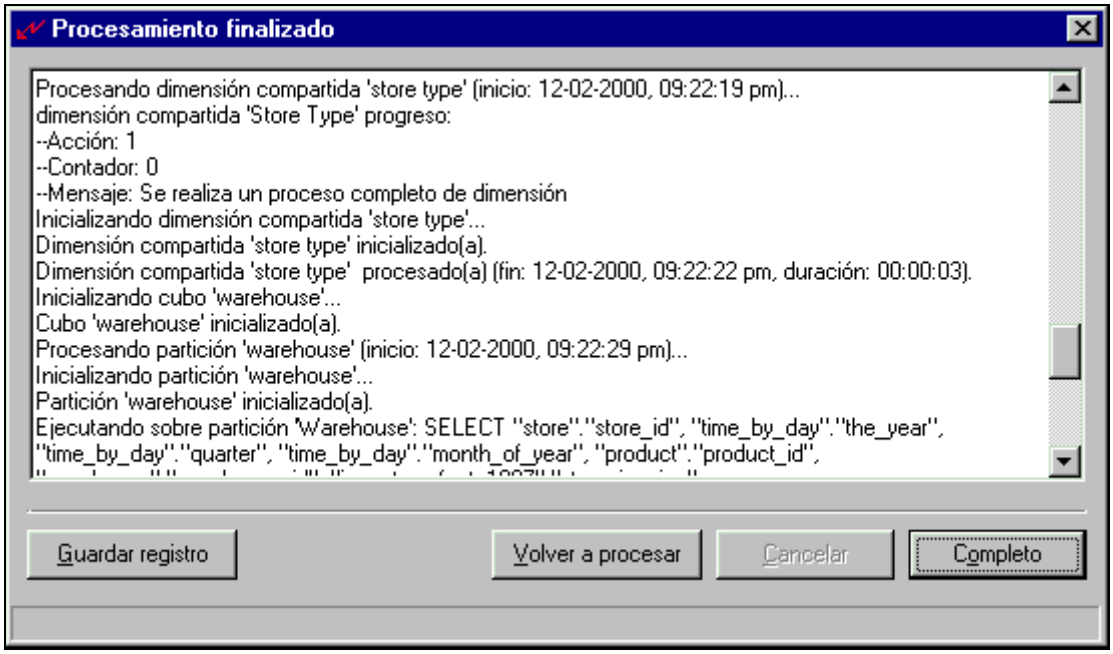


En dicho formulario, se debe optar por la opción *Importar* si se desean importar los cubos base, y *Omitir* si se prefiere importar solo el cubo virtual.
Para cada cubo base, en caso de que la base de datos vinculada se encuentre en un servidor Microsoft SQL Server, se proporcionará un formulario como el citado en el punto 3.

Si se da el caso de que dos o más cubos base tengan el mismo origen de datos, solo se proporcionará la opción de migrar la base de datos por uno de ellos.

En el último paso del proceso de importación se brinda la posibilidad de procesar el cubo que se acaba de generar (tanto sea virtual o regular).

Procesamiento



Durante el procesamiento del cubo, se despliega una traza de las acciones llevadas a cabo por el servidor. Una vez finalizado el proceso, presionando el botón *Guardar registro*, se puede almacenar esta información generada en un archivo de texto.

El procesamiento del cubo puede ser repetido tantas veces como sea necesario presionando el botón *Volver a procesar.*

Cabe señalar que al realizar la importación se recrean en el servidor destino todos los elementos de la metadata del cubo a excepción de los roles.

7. Generar archivo para ser visualizado en un browser

Esta operación lleva la metadata de un cubo a un archivo HTM, de forma de que ésta pueda ser visualizada luego en un browser. La realización de esta operación requiere que el cubo se haya exportado previamente.

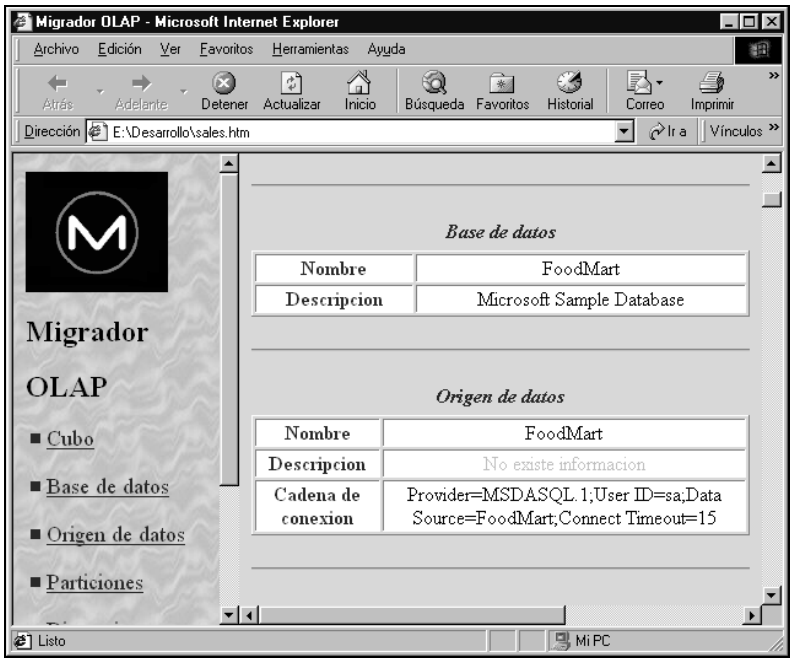
Para realizar esta acción se deben seguir los siguientes pasos:

1. Seleccionar un cubo que ha sido exportado en la interfaz de la aplicación.
2. Accionar el botón *Generar un archivo para visualización*. Si el cubo sobre el cual se está aplicando la operación tiene cambios que no han sido guardados, se presentará un cuadro de diálogo que indica que el cubo debe ser guardado antes de realizar la acción. Si el cubo no tiene cambios a ser guardados se desplegará un formulario que le permitirá especificar la ubicación del archivo a generar.

Cuadro en el que se especifica donde guardar el archivo HTM

3. Si el archivo se generó con éxito, se presentará la opción de visualizar el archivo recién creado en el browser predeterminado.

Archivo generado visto a través de un browser



8. Eliminar un elemento de un cubo

Esta operación permite eliminar elementos de la metadata de un cubo. La diversidad de elementos con posibilidad de eliminar depende del hecho de si se está trabajando sobre un cubo regular o virtual. Si se trata de un cubo regular, todos los elementos a excepción de la base de datos y el origen de datos podrán ser eliminados. Con la restricción de que no se puede eliminar la última dimensión o medida de un cubo, así como el último nivel de una dimensión. En el caso de los cubos virtuales, los elementos que no pueden ser eliminados son: base de datos, origen de datos, cubos base, niveles, propiedades miembro y medidas. Teniendo los cubos virtuales la misma restricción que los cubos regulares acerca de la eliminación de la última dimensión.

A continuación se listan algunos efectos secundarios que pueden traer aparejados la eliminación de un elemento:

- Si el elemento que se está eliminando es una medida, una dimensión, un nivel regular o una propiedad miembro, su eliminación involucra la eliminación también de todas las agregaciones del cubo.

- Si el elemento a eliminar es una partición, se eliminan automáticamente todas las agregaciones de dicha partición.

- La eliminación de una medida, una dimensión (regular o virtual), un nivel o una propiedad miembro, puede dejar inválidos miembros calculados.

La eliminación se puede realizar de la siguiente manera:

1. Seleccionar el elemento que se desea eliminar.
2. Accionar la opción de eliminar el elemento, esta se puede activar desde la barra de tareas, el menú *Acciones*, o bien del menú contextual correspondiente.
3. Confirmar la acción.

9. Renombrar un elemento de un cubo

Haciendo uso de esta operación se pueden renombrar los elementos de la metadata de un cubo. La diversidad de elementos con posibilidad de renombrar depende del hecho de si se está trabajando sobre un cubo regular o virtual. En el caso de cubos regulares todos los elementos de su metadata pueden ser renombrados. Para cubos virtuales, los únicos elementos que se admite renombrar son las medidas, el origen de datos, la base de datos y el propio cubo.

Renombrar un elemento de la metadata de un cubo puede ocasionar los siguientes efectos secundarios:

- Si el elemento renombrado es un cubo regular, una medida, una dimensión, un nivel regular o una propiedad miembro, el renombramiento involucra la eliminación de todas las agregaciones del cubo.

- Si el elemento a renombrar es una partición, se eliminan automáticamente todas las agregaciones de dicha partición.

- Renombrar un cubo (regular o virtual), una medida, una dimensión (regular o virtual), un nivel (regular o virtual), o una propiedad miembro, puede dejar inválidos miembros calculados.

La acción de renombrar se puede realizar de las siguientes tres formas:

1. Desde el formulario de propiedades del elemento correspondiente.
2. Haciendo un click simple sobre el elemento que se desea renombrar (tanto en el panel derecho como en el izquierdo).
3. Desde el menú contextual del elemento a renombrar.

10. Cambiar descripción de un elemento de un cubo

Mediante esta operación es posible cambiar la descripción de los elementos de la metadata de un cubo (o agregar una a los que no la posean).

La acción de cambiar la descripción solo se puede llevar a cabo a través del formulario de propiedades del elemento correspondiente. Por lo tanto, para realizar este procedimiento, se debe abrir el formulario de propiedades y especificar desde allí la nueva descripción.

11. Especificar si una dimensión es privada o compartida

Dada una dimensión de un cubo, esta operación permite transformarla a privada, en caso de que sea compartida, o llevarla a compartida en caso de que sea privada. Esta operación solo se podrá llevar a cabo si se está trabajando sobre un cubo regular.

Para efectuar la operación se debe abrir el formulario de propiedades de la dimensión correspondiente, para luego especificar en el mismo si se desea que la dimensión sea privada o compartida.

Cabe destacar, que al cambiar una dimensión de compartida a privada, o de privada a compartida, se eliminan todas las agregaciones del cubo.

12. Especificar si una medida es oculta o no

Dada una medida de un cubo, haciendo uso de esta operación se podrá especificar si se desea que la medida sea oculta o no. Esta acción se permite realizar si se está trabajando sobre un cubo regular y solo se puede efectuar desde el formulario de propiedades de la medida correspondiente.

13. Especificar el modo de almacenamiento de una partición

Con esta operación es posible indicar el modo de almacenamiento deseado para una partición, pudiendo ser éste MOLAP, HOLAP o ROLAP.

Este cambio del modo de almacenamiento solo es posible desde el formulario de propiedades de la partición correspondiente.