Revisión de Cubos de datos con Excepciones, Implementación y análisis.

Gustavo Alfredo Guarino
L.U.: 1036/87

Director: Dr. Alejandro Vaisman
Co-Director: Lic. Mauricio Minuto Espil

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Marzo de 2003

Abstract

Revising Data Cubes with Exceptions: implementation and analysis.

Gustavo Alfredo Guarino

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Marzo de 2003

Data Warehouses are complex systems that integrate and store data with high levels of aggregation for helping in the decision-making process. Nowadays the most of them use the multidimensional model for data representation. This model, although adequate for the general case, fails to capture some real world situations, for example, when exceptions occur. In an insurance company, for example, the general rule is that all the clients can be considered *Reliable*, if they are between forty and fifty years old; but in a specific moment of the cycle of life of the Data Warehouse, because of business policies, an exception is established for those who have more than one fine for high-speed driving; at present, this situation cannot be addressed using the existing models, without having to change the implementation of the Data Warehouse. Thus, it is interesting to study under which conditions previous proposals for revision of dimensions, and recalculation of data cubes of multidimensional databases, are feasible and scalable.

Our proposal implements the algorithms proposed by Minuto and Vaisman for allowing the definition of exceptions in a multidimensional space. We develop the interface and the components allowing the definition and execution of revisions and creating new revisions on the different dimensions of the cubes; thus, revision of the dimensions and of the multidimensional cubes are defined, in order to study the feasibleness in different implementation scenarios, developing testing under different architectures of access to the information of the cubes, and analyzing quantitatively the scalability of the proposed algorithms.

Resumen

Revisión de Cubos de datos con Excepciones, Implementación y análisis.

Gustavo Alfredo Guarino

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Marzo de 2003

Los Data Warehouses son sistemas complejos que integran y almacenan datos con altos niveles de agregación para ayudar en el proceso de toma de decisiones. Hoy en día la mayoría de los Data Warehouses utilizan el modelo multidimensional para la representación de los datos. Este modelo, aunque es adecuado para el caso general, falla al querer abarcar algunas situaciones particulares del mundo, como en el caso que ocurran excepciones. Por ejemplo, en una compañía de seguros, la regla general es que todos los clientes se consideren *Confiables*, si tienen entre cuarenta y cincuenta años de edad; pero en un momento específico del ciclo de vida del Data Warehouse, debido a políticas de negocio, se establece una excepción para los que tengan más de una multa por exceso de velocidad; actualmente, esta situación no se puede tratar usando los modelos existentes, sin tener que modificar la implementación del Data Warehouse. Por este motivo, es interesante estudiar las propuestas existentes acerca de revisión de dimensiones y de recálculo de cubos de datos de bases de datos multidimensionales, y establecer bajo que condiciones son factibles y escalables.

Nuestra propuesta implementa los algoritmos propuestos por Minuto y Vaisman para permitir la definición de excepciones en un espacio multidimensional. Desarrollamos la interfaz y los componentes permitiendo la definición y ejecución de revisiones y creando nuevas revisiones en las diferentes dimensiones de los cubos; de este modo, definimos la revisión de las dimensiones y de los cubos multidimensionales, para estudiar la factibilidad de su ejecución en diferentes escenarios, desarrollando pruebas bajo diferentes arquitecturas de acceso a la información de los cubos, y analizando cuantitativamente la escalabilidad de los algoritmos propuestos.

Agradecimientos

Quiero agradecer a las personas que me ayudaron y apoyaron para que este trabajo sea posible. Principalmente a los docentes que me dieron la posibilidad de desarrollar este trabajo, el Dr. Alejandro Vaisman y el Lic. Mauricio Minuto Espil, que en todo momento estuvieron para guiarme. A mi familia, en especial a mi madre, y a aquellas personas que estuvieron muy cerca de mí durante el desarrollo de este trabajo, apoyándome con paciencia durante las largas horas de estudio y dándome fuerzas para llegar al final esperado.

Por ultimo, quiero dedicar este esfuerzo y esta tesis a la memoria de mi Padre, el que me dio las últimas fuerzas para seguir cuando parecía que no era posible.

Índice

Introducción

Conceptos generales sobre Data Warehouses

Los Data Warehouses, son sistemas complejos que integran y almacenan datos con altos niveles de agregación para ayudar en la toma de decisiones. Para ello existe una tecnología denominada OLAP (On-line Analytical Processing), que permite transformar los datos transaccionales permitiéndole al usuario final realizar un análisis Multidimensional. El Análisis Multidimensional es el proceso de analizar los datos organizados de acuerdo a variables (por ejemplo producto, región, fechas, etc.) de modo que quien debe tomar decisiones pueda encontrar la mayor utilidad posible en la información almacenada [WB97], [Wid95].

Para diferenciar OLAP de OLTP (On-line Transaction Processing), decimos que OLAP utiliza una clase de servidores de bases de datos que están diseñados para permitir acceso y análisis de los datos según la necesidad del usuario, mientras que las transacciones residen en bases de datos relacionales o en otro tipo de archivos.

Las bases de datos relacionales están organizadas en listas de "registros", cada grupo de listas constituyen una Tabla, cada "registro" contiene información relativa al mismo, la que está organizada en "campos". Esta estructura no se presenta como la más adecuada para un análisis multidimensional, debido a esto surgieron las bases de datos multidimensionales.

Muchos trabajos han propuesto modelos para aplicaciones OLAP y muchos de estos basados en la idea original del Star Schema, en donde los datos son almacenados en un conjunto de tablas, dimensiones y tablas de datos.

Gracias a esta estructura los sistemas OLAP soportan el análisis de los datos, en dicha estructura los datos están representados como un conjunto de dimensiones y métricas. Las dimensiones usualmente están organizadas en jerarquías, dichas jerarquías soportan distintos niveles de agregación. Por su parte las métricas pueden verse como puntos pertenecientes a un espacio multidimensional.

Normalmente una jerarquía de dimensiones es representada por un grafo acíclico dirigido, con un único nivel inferior y un nivel superior distinguido llamado [All]. Las extensiones para cada agregación en la jerarquía, llamada instancia de la dimensión, son representadas por funciones sobre las instancias de los niveles, estas son llamadas funciones roll-up y sólo se proveen funciones roll-up entre niveles adyacentes, mientras que funciones roll-up entre niveles no adyacentes se computan mediante la composición de funciones. [HMV99a] y [HMV99b].

A continuación veremos como se catalogan los distintos tipos de Data Warehouses, los modelos de construcción y algunas de sus características.

Durante el proceso de construcción de un Data Warehouse, los cambios en los esquemas de las fuentes de información, afectan mucho a los Data Warehouses. Además de los problemas de cambios en los esquemas, los Data Warehouses presentan el problema adicional de mantener la información actualizada, para que realmente reflejen la realidad de la empresa y sirvan como herramienta de decisión [Vas00].

Según [Inm92] "Un Data Warehouse es una colección de datos subject-oriented, integrated, time-variant y non-volatile para ayudar al proceso de toma de decisiones gerenciales".

Subject Oriented: datos que brindan información de un sujeto del negocio, en lugar de concentrarse en las transacciones.

Integrated: se integran datos de distintas fuentes para dar una visión de un todo coherente.

Time-Variant: todos los datos se asocian con un período de tiempo específico.

Non-Volatile: los datos son estables, se agregan datos pero no se borran.

Esta es una definición muy pura, luego fueron surgiendo conceptos como los de Data Marts y Data Warehouses corporativos.

Como citamos de [Kim92], un Data Warehouse es una copia de los datos transaccionales específicamente estructurados para consultas y análisis.

Básicamente, para guardar estas estructuras, los datos pueden almacenarse en una RDBMS (Relational DataBase Management System) o en una Base de datos Multidimensional, estas últimas por lo general son bases propietarias y se alimentan de bases relacionales. En general los Data Warehouses son Read Only y se los identifica como ambientes OLAP (On-Line Analytical Procesing) en contraposición con los OLTP (On-Line Transactional Procesing) que son transaccionales.

Metodologías de construcción

Para la construcción de una Data Warehouse se han presentado varias metodologías, citaremos a algunas de ellas [ZR99], [Fir98]:

TOP Down: en este modelo se construye el Data Warehouse con la información completa e histórica (en general en una base relacional) y los Data Marts se arman en base a este, pero con información menos detallada, o sea sumarizada. Los Data Marts usarán el Star Schema [Kim96], para mejorar la performance. Este esquema se basa en la idea de ir definiendo jerárquicamente dimensiones con sus niveles y métricas.

De esta manera es muy costoso implementarlos, ya que hay que conocer todos los requerimientos de toda la organización para definir primero el Data Warehouse Corporativo

y luego podemos hacer los Data Warehouses departamentales o Data Marts. En este caso existe un metadata compartido, pero nunca es fácil llegar a su conclusión.

Bottom Up: en este caso partimos de los Data Marts, pero con toda la información de detalle, estos serán los bloques para construir todo el Data Warehouse. Aquí no hay un metadata compartido por todos los Data Marts. No es escalable como la Top-Down, pero es más rápido obtener resultados, es un enfoque más realista.

Enterprise Data Mart Architecture (EDMA): este modelo se basa en el Bottom Up, pero intenta hacerlo incremental y escalable. Para esto usa un GMR (Global Metadata Repository) esto servirá para compartir el modelo lógico, pero puede no ser compartido físicamente.

DataStage / DataMart Architecture (DS/DMA): este es similar al anterior, EDMA, pero con la diferencia que no se construye un Data Warehouse físico, sino que el Data Warehouse es la conjunción de todos los Data Marts. Es como si fuera un Warehouse Corporativo Abstracto, este esquema asume que todos los parámetros están definidos en los Data Marts y no hay nada a nivel Empresa, por esto último parece no ser muy real, ya que en la mayoría de las empresas hay información compartida que es bueno que este en un sólo lugar.

Distributed DW / DataMart Architecture (DDW / DMA): es similar a EDMA pero con dos diferencias: provee una capa lógica de base de datos para mapear el modelo lógico a las tablas físicas de los Data Marts, y segundo, provee una forma transparente y común para todos los Data Marts de acceder a los datos de la capa lógica definida.

Distributed Knowledge Management Architecture (DKMA): este modelo es como si le agregara una capa Orientada a Objetos sobre la definición de EDMA, por medio de la definición de objetos, clases abstractas y utilizando las facilidades de la herencia, se simplifica el reuso del esquema.

Distintas arquitecturas de Data Warehouses

Algunas características de las distintas arquitecturas de Data Warehouses son:

Sobre bases Multidimensionales (MOLAP Multidimensional On-Line Analytical Procesing)

- Tiempo de respuesta bajo, esta todo precalculado.
- Si se pregunta algo que no esta precalculado no funciona o puede demorar horas en calcular.
- Diseño propietario.
- Se alimenta de una RDBMS.
- Lenguaje de consulta propietario.

- Flexibilidad limitada.
- Un cambio en el negocio implica recalcular todos los cubos.
- Ineficientes cuando se crece en volumen y dimensiones.
- Crecimiento exponencial ante cualquier agregación.
- Reducido potencial de almacenamiento.
- Reducido potencial en número de dimensiones.
- Sirve para productos pequeños y estáticos.
- Sirve cuando el tiempo de respuesta es crítico.
- Sirve cuando hay poco volumen de datos.
- Sirve para hacer análisis de alto nivel, sin detalle.

Sobre bases Relacionales (ROLAP Relational On-Line Analytical Procesing)

- Tiempo de respuesta de segundos a minutos.
- Se puede analizar todo lo que hay en el Data Warehouse.
- Distintos motores de bases de datos.
- Acceso mediante lenguaje SQL.
- Flexible y escalable.
- Se pueden trasladar los cambios del negocio al Data Warehouse.
- No se deben regenerar cubos.
- Formato relacional.
- No cae cuando aumenta el volumen o cantidad de dimensiones.
- No hay límite en la capacidad de almacenamiento.
- No hay límite en la capacidad en cantidad de dimensiones.
- Sirve para modelos dimensionales grandes y dinámicos.
- Sirve para grandes volúmenes.
- Sirve para hacer análisis a nivel transaccional.

Un híbrido (HOLAP Hybrid On-Line Analytical Procesing)

Es cuando se usa MOLAP y no se puede resolver una consulta, entonces se accede a la base de datos relacional, pero esto no es recomendable, las consultas no están optimizadas y el tiempo de respuesta puede ser grande.

Introducción al problema de las Excepciones

De los modelos presentados, enfocaremos nuestro estudio sobre el modelo MOLAP, ya que es el más adecuado para estos casos, donde es importante la previsión de los tiempos de respuesta.

El modelo multidimensional es adecuado para el almacenamiento de cualquier Data Warehouse en general, dando al usuario final mucha potencia a la hora de hacer un análisis de la situación actual de la empresa y con buena performance. Pero detectamos el problema de que al querer abarcar situaciones del mundo real donde ocurren excepciones, este

modelo evidencia algunos problemas. Estos problemas no son inherentes al modelo MOLAP, sino a cualquier Data Warehouse en general. Enfocaremos el estudio sobre el modelo MOLAP, porque es el más adecuado por sus características.

El problema al que nos referimos, es el caso donde se quieran expresar situaciones en las que ocurren excepciones, un ejemplo simple de esta situación es:

Supongamos una compañía de seguros, donde todos los clientes pueden ser considerados *Confiables*, si tienen entre 40 y 50 años, esta es una regla general y definida al momento de diseñar el Data Warehouse. Ahora supongamos que en algún momento del ciclo de vida del Data Warehouse, por políticas de la empresa o cualquier otra circunstancia, se decide agregar una excepción a esta condición, por ejemplo, exceptuar la asignación de la categoría *Confiable* a todos aquellos clientes que tengan más de una multa por exceso de velocidad y asignarle la categoría *Dudoso*. Como esta excepción surge una vez que el Data Warehouse se encuentra en funcionamiento, para realizar la modificación, deberíamos tocar el diseño del mismo y tal vez reprocesar muchas cubos, de esta manera estaríamos afectando a muchos usuarios por un motivo que tal vez es de interés para algún usuario particular. Por lo que podríamos concluir que no es manejable de manera aceptable con los modelos existentes en la actualidad, sin tener que modificar el modelo.

Nuestro enfoque

Para permitir un manejo más eficiente de situaciones como la presentada en el ejemplo del párrafo anterior, es que [MV01], [MV02] y [MV03] proponen un modelo de definición de excepciones, que permita manejar estas situaciones particulares sobre un cubo, sin sacarlo de línea. Además que permita ver y trabajar sobre las consecuencias de aplicar la excepción, sin necesidad de afectar a los usuarios que quizás no estén interesados en el cambio.

Veremos un poco más en detalle porque creemos necesario que exista un modelo de definición de excepciones sobre cubos multidimensionales. Sabemos que la información almacenada en un Data Warehouse, no siempre refleja de manera exacta los distintos índices y datos dentro de una organización. En la práctica los datos deberían verse como una representación aproximada de la realidad, por lo cual debería existir la posibilidad de contemplar casos particulares, situaciones imprevistas o no consideradas en la implementación actual.

Por eso proponemos un método de revisión de datos a fin de permitir la realización de un análisis mas acertado. Hasta el momento no hay implementaciones de algoritmos de revisión de cubos, desde el punto de vista presentado en [MV02], por eso trabajaremos en la implementación para ir acercándonos al desarrollo de un componente y herramienta destinadas a la definición de excepciones sobre cualquier Data Warehouse.

Organización de la tesis

A lo largo de esta tesis presentaremos los siguientes puntos:

- Relevamiento general sobre Data Warehouses, beneficios y problemas. En particular estudiamos el modelo Multidimensional, ya visto en esta sección.
- Ambientes OLAP y definición de interfaces de los componentes y estructuras que servirán para almacenar los datos temporales que serán utilizados durante la ejecución de los algoritmos.
- Desarrollo del componente de revisión. Este permite la definición de excepciones, revisión de dimensiones y cubos, incluyendo el recálculo de las agregaciones y emisión de resultados.
- Desarrollo de la interfaz de usuario para facilitar la creación de las revisiones y una capa de abstracción sobre el modelo de objetos de la base de datos multidimensional a utilizar.
- Realización de testeos y análisis de comportamiento de los algoritmos bajo distintas condiciones. Analizaremos las consecuencias de los distintos grados de selectividad de las reglas y en que casos es viable la revisión.

En el Capitulo 1 presentaremos una introducción a los ambientes OLAP, métodos de acceso a los datos multidimensionales y las herramientas seleccionadas. En el Capitulo 2 veremos más detalles de que son las excepciones, ejemplos en donde son aplicables y estimación de sus beneficios. En el Capitulo 3 veremos una descripción resumida de los algoritmos desarrollados en los componentes de revisión, las estructuras de datos básicas utilizadas para la revisión, un ejemplo de revisión de un cubo y por último el modelo de objetos definido en los componentes.

En el Capitulo 4 haremos una presentación de los casos de prueba realizados con el componente de revisión, y haremos un análisis cuantitativo de cotas bajo las cuales los algoritmos demuestren un comportamiento razonable. También se harán análisis de tiempos y estimaciones de utilización.

Finalmente en el Capitulo 5 presentaremos las conclusiones generales de este trabajo, contribuciones y futuros trabajos.

Capítulo 1 Introducción a los ambientes OLAP

1.1 Software utilizado

Para la extracción de los datos que nos permitan llegar a una conclusión acerca de cuando es beneficiosa y factible la aplicación de un algoritmo de revisión de cubos sobre nuestro Data Warehouse, se desarrollaron dos algoritmos. El primero es propuesto por [MV01] y trata sobre la revisión de dimensiones, el segundo es propuesto en [MV02]. En dicho paper se presentan los algoritmos desglosados en tres partes para facilitar su entendimiento, pero aquí los integramos en un sólo componente que ejecutará del lado del servidor para optimizar el tráfico en la red y porque es el lugar natural para ejecutar lo más unido posible a la herramienta de OLAP.

Para el desarrollo del componente y la realización de los diferentes tests, se seleccionó una base de datos que brinde servicios de OLAP, en nuestro caso Microsoft SQL Server 2000 Enterprise Edition con el adicional del módulo de OLAP del SQL Server denominado, Análisis Services, debido a que es una de las bases de datos más populares del mercado.

Por otro lado, se debía optar por el lenguaje con el cual sería desarrollado el componente. Este lenguaje debía tener como características, que fuera potente y capaz de generar código óptimo, a la vez que tenga una interfaz natural con los objetos del servidor de base de datos. Para este desarrollo seleccionamos Borland Delphi 5. [BDel5], [Bru99], [HTTPDJ], [HTTPMDX], [Mat02], [MSHLP99], [MSSQL2k], [MSSQLAS], [You99].

Para presentar de qué manera el componente interactúa con el servidor OLAP, explicamos de manera breve el modelo de objetos que propone Microsoft para acceder a los datos multidimensionales.

1.2 Conceptos básicos de OLAP

En la próxima sección haremos un recorrido por los conceptos básicos del modelo OLAP.y veremos en general, las distintas formas de acceder a los servicios de OLAP, tanto del lado del cliente como del lado del servidor.

Esquemas multidimensionales

El objeto central de los esquemas multidimensionales es el Cubo, el cual consiste de un conjunto estructurado de dimensiones (dimensions), jerarquías (hierarchies), niveles (levels), miembros (members) y métricas (measures).

En OLE DB for OLAP, el cubo es un concepto básico, en SQL Server existen tres tipos de cubos distintos: Cubos Virtuales, Hipercubos y Multicubos.

Hipercubos y Multicubos

Las bases de datos multidimensionales pueden exponer sus datos utilizando dos tipos de cubos: *hipercubos* y *multicubos*. En el modelo de hipercubos, todos los datos aparecen en un sólo cubo, todos los datos del esquema están accesibles, y todas las dimensiones están en el cubo.

En el modelo de Multicubos, los datos están segmentados en distintos cubos más pequeños, cada cubo esta compuesto por un subconjunto de las dimensiones. Básicamente ambos difieren en la información del metadata que tienen accesibles desde cada cubo. En un hipercubo cada dimensión pertenece sólo a un cubo, mientras que en un Multicubo una dimensión puede ser parte de muchos cubos.

En un sistema Multicubo, las dimensiones existen en el esquema y los diferentes cubos utilizan un subconjunto de estas dimensiones. En contraposición, en un sistema de Hipercubo, cada hipercubo tiene su conjunto independiente de dimensiones.

Dimensiones

Una Dimensión es una categoría independiente de dato, en general derivan de las entidades definidas en el negocio particular y de los tipos de datos a modelar en la base de datos. Una dimensión en general contiene ítems que servirán para filtrar las consultas que se hagan para seleccionar los distintos valores (métricas) de la base de datos.

Por ejemplo, haciendo referencia a la base de datos FoodMart, que viene como ejemplo en Microsoft SQL Server 2000 Analysis Services, tomamos el cubo conteniendo información de las ventas (Sales), y tomamos cuatro de las dimensiones definidas, Store, Time, Customers, Product.

Por ejemplo la dimensión Customers tiene los siguientes miembros de último nivel, {Robert Damstra, Rebecca Kanagaki, Kim Brunner, Brenda Blumberg}.

Jerarquías

Una Jerarquía es el camino de agregación de una dimensión. Una dimensión puede tener múltiples niveles de granularidad, que tienen entre ellas relaciones padre-hijo. Una jerarquía define como están relacionados estos niveles. Las jerarquías definen de que manera los datos pueden ser agrupados.

En el ejemplo de la dimensión Customers sería:

All Customers
  Country
    State Province

City
Name

Niveles

Un Nivel es un escalón de agregación dentro de la Jerarquía. En el ejemplo anterior, cada escalón de la jerarquía es un Nivel.

All Customers = {All}
Country = {Canada, USA, Mexico}
State Province = {Oaxaca, WA, CA, OR}
City = {Los Angeles, Downey, Corvallis, Bremerton, Edmonds, Arcadia}
Name = {Robert Damstra, Rebecca Kanagaki, Kim Brunner, Brenda Blumberg, Forham Kenton, Malik Brittany, Eldridge Jeannette}

Miembros

Un Miembro es un ítem (dato) en una dimensión. Es para describir los valores de la base de datos. Los miembros de las hojas de la jerarquía no tienen hijos, los miembros del nivel superior no tienen padre y todos los demás tienen por lo menos un padre y un hijo.

{All} (es padre de) {USA, Canadá}
{USA} (es padre de) {CA}
{CA (es padre de) {Kenton Forham}

Fact Table

Los Data Warehouses están basados en uno o más Fact tables, que son las tablas que guardan la información numérica del negocio, esta información será la que luego se sumarizará para brindar información histórica.

Métricas

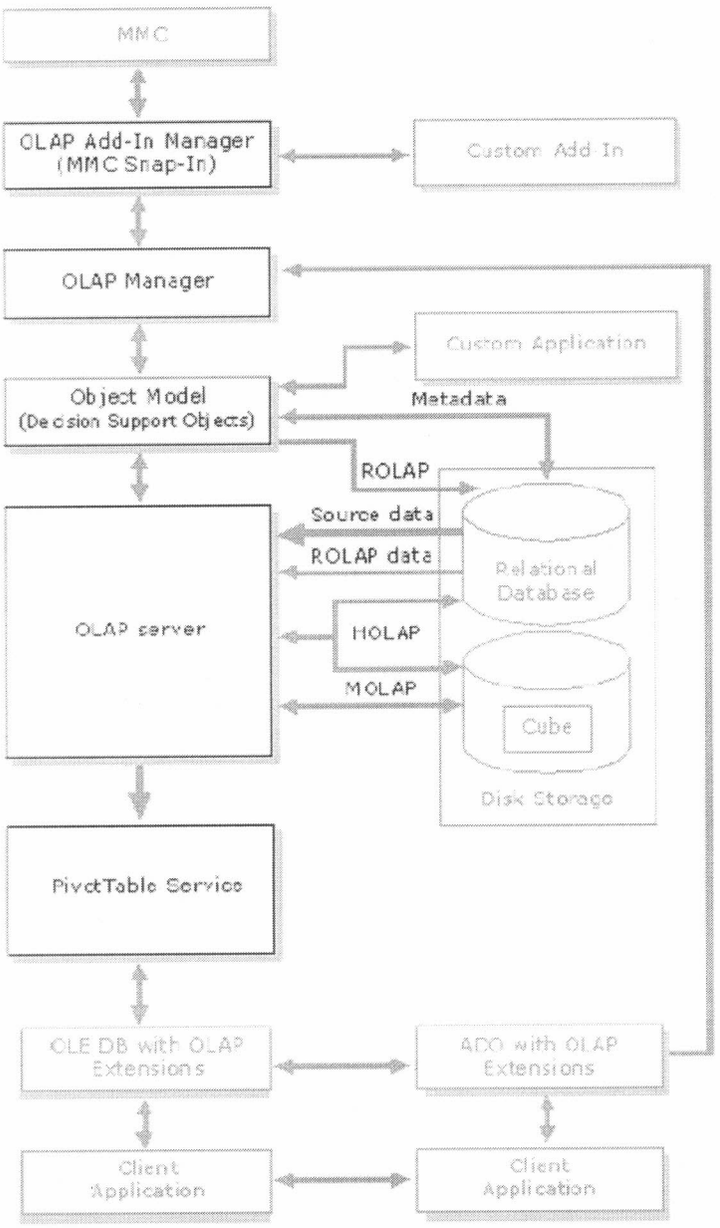
En un cubo, una métrica es un valor basado en una columna del Fact Table y en general son numéricas. Son los valores centrales para realizar el análisis sobre un cubo. Ejemplos de métricas pueden ser, *Monto de ventas* o *Cantidad de unidades vendidas*. Por cada métrica en un cubo, cada celda tiene un valor.

Paths

Llamaremos Paths a los caminos entre los distintos niveles de una jerarquía. Un Path esta formado por un valor en cada nivel de la jerarquía, con funciones que los relacionan uno a uno.

1.3 Arquitectura de los servicios de OLAP

En el siguiente diagrama presentamos los distintos componentes y funciones del Microsoft Analysis Server 2000.



MMC: Microsoft Management Console, es la consola estándar de operación de las aplicaciones de administración para el usuario.

OLAP Manager: también llamado Analysis Manager en SQL Server 2000, es una aplicación que provee una interfaz para que el usuario acceda a los datos del Analysis Server. Algunas de sus funciones son:

- Administración del Analysis Services
- Crear base de datos y setear las fuentes de datos
- Construir y procesar cubos
- Especificar opciones de almacenamiento y hacer optimizaciones
- Manejar la seguridad
- Visualizar datos, dimensiones y otros objetos.

Object Model (Decision Support Objects): es el modelo de objetos para acceder directamente a las funciones de OLAP Server, como ser crear cubos, dimensiones, procesar, etc.

Custom Application: cualquier aplicación que acceda a los objetos del DSO. El componente de revisión los utiliza para crear y procesar el cubo resultante de la revisión.

Metadata: aquí se almacena toda la información referente a la definición de los cubos, dimensiones, jerarquías, etc. A este metadata accede el componente de revisión para obtener la información de las dimensiones.

Cube Storage: sirve para definir las fuentes de datos, el modelo y lugar donde se almacenarán los datos.

Relational Database: dependiendo del modelo del cubo (ROLAP, HOLAP), estarán los datos fuentes.

Cube: aquí se almacenarán las agregaciones en el caso de cubos MOLAP y HOLAP. Con respecto al tipo de cubo a generar, las opciones son MOLAP, ROLAP y HOLAP, cuyas características se vieron en la Introducción.

OLAP Server: es el motor del Analysis server.

PivotTable Service: es un proveedor de OLE DB para datos multidimensionales y operaciones de datamining. Provee soporte para un subconjunto de las sentencias del lenguaje SQL y para sentencias MDX. El componente PivotTable Service permite que las aplicaciones obtengan datos de tablas o datos multidimensionales.

OLEDB with OLAP: es el proveedor de OLEDB para servicios de OLAP de SQL Server 2000.

ADO with OLAP: también llamado ADO MD, contiene extensiones de ADO, que son nuevos objetos para acceder a funcionalidades específicas del modelo OLAP. Algunos de estos objetos son Catalog y CellSet.

Client Application: cualquier aplicación que acceda a los objetos del OLEDB with OLAP y ADO with OLAP. Aquí los utilizamos para acceder a los metadatos y a los datos de los cubos, tanto desde el componente que correrá en el servidor como desde la aplicación cliente.

1.4 Métodos de acceso a datos y metadata del OLAP Server

Los datos almacenados en cubos creados con Microsoft SQL Server 2000 OLAP Services, son accesibles vía el *Microsoft OLE DB Provider for OLAP Service, que es el driver de OLE DB para OLAP*. Una de las formas de obtener los datos del metadata es utilizando el método llamado OpenSchema que pertenece al objeto Connection de ADO (ActiveX Data Objects), la otra opción es accediendo vía los objetos definidos por Microsoft que son Catalog y Cellset.

OLE DB for OLAP es un conjunto de interfaces COM para extender OLE DB y de esta manera acceder de un forma eficiente a los datos multidimensionales almacenados en cubos de SQL Server Analysis Services. Primero veremos como acceder utilizando el método OpenSchema, seteando los distintos parámetros podremos obtener información de Cubos, Dimensiones, Jerarquías, Niveles, Miembros, Propiedades de los miembros y Métricas.

Uno de estos parámetros, llamado Criteria, nos permite filtrar la información que queremos obtener, por ejemplo, si queremos obtener información de una Dimensión particular, deberemos setear la condición correspondiente.

Ejemplo:

En el siguiente ejemplo traeremos todos los miembros de un determinado nivel, de una dimensión de un cubo.

```
Crit:=VarArrayOf([DataSourceName,Null,CubeName,DimensionName,  DimensionName,LevelName]);
VSRs:= VSConn.OpenSchema(aObject,Crit, IfProviderSpecific);
```

aObject: puede contener los siguientes valores, AdSchemaCatalogs, AdSchemaCubes, adSchemaDimensions, adSchemaHierarchies, adSchemaLevels, adSchemaMeasures, adSchemaMembers y adProviderSpecific, los que determinarán que tipos de objetos retornará el método.

IfProviderSpecific: en general el parámetro va en blanco, salvo que el primer parámetro sea adProviderSpecific, en este caso se indicará el GUID del provider.

El resultado quedará almacenado en un Recordset de ADO, el cual se podrá recorrer para obtener los distintos valores.

Con el método OpenSchema obtenemos información del metadata, pero no información de los datos, para ello veremos las expresiones MDX, que es un lenguaje similar al SQL, que servirá para hacer consultas sobre los cubos.

1.5 Introducción a las sentencias MDX

La sigla MDX significa Multidimensional Expressions, y es un lenguaje de consultas con una sintaxis similar a la de SQL (Structured Query Language). No es una extensión de SQL, ya que algunas de las funciones de MDX pueden suplantarse por sentencias SQL, pero no de manera eficiente.

Básicamente una sentencia MDX, consta de un SELECT, un FROM y un WHERE, además de contar con varias funciones muy potentes para obtener solamente la información que se busca. A continuación veremos su sintaxis básica.

SELECT axis_specification ON COLUMNS,
axis_specification ON ROWS
FROM cube_name
WHERE slicer_specification

axis_specification: puede ser pensado como una selección de miembros para ese eje, un miembro es un ítem en una dimensión o métrica.

***cube_name**:* es el nombre del cubo a consultar.

slicer_specification: en la cláusula WHERE es opcional, si no es especificada el resultado retornará la métrica default del cubo.

A continuación mostramos algunos ejemplos de consultas MDX, utilizando la base de datos de ejemplo Foodmart del OLAP Server:

Pagos por año para todos los países

SELECT {[Date].children} ON COLUMNS, ([cust].children) ON ROWS
FROM Payment_Cube
WHERE Payment

Pagos por cuatrimestre en 1997 para todos los países

SELECT {[Date].[Year].[1997].children} ON COLUMNS, ([cust].children) ON ROWS
FROM Payment_Cube
WHERE Payment

Pagos por año para todos los clientes de Alemania

SELECT {[Date].[Year].members} ON COLUMNS,
([cust].[Country].[Germany].children) ON ROWS FROM Payment_Cube
WHERE Payment

Ahora veremos algunas de las funciones más comunes utilizadas en las sentencias MDX.

Members

La forma más simple de un axis_especification, es obtener los MEMBERS de un determinado nivel de una dimensión, incluyendo la dimensión especial Measures.

SELECT Measures.MEMBERS ON COLUMNS,
[customers].MEMBERS ON ROWS
FROM [Sales]

También se puede seleccionar un miembro particular de una dimensión.

SELECT Measures.MEMBERS ON COLUMNS,
{[customers].[Country].[mexico],
[customers].[Country].[canada]} ON ROWS
FROM [Sales]

Children

Si quisiéramos obtener los miembros hijos de un miembro específico. Si en el caso anterior quisiéramos obtener los miembros que pertenecen a México y Canadá, escribiríamos:

SELECT Measures.MEMBERS ON COLUMNS,
{[customers].[Country].[mexico].CHILDREN,
[[customers].[Country].[canada].CHILDREN} ON ROWS
FROM [Sales]

Descendants

Esta función nos permite hacer un drill-down en la jerarquía, pasar a un nivel inferior y devolver los miembros que se indican en la variable *flags*.

DESCENDANTS(*member*, *level* [, *flags*])

SELECT {[Measures].[Unit Sales]} ON COLUMNS,
(DESCENDANTS([customers].Canada, [State Province])) ON ROWS
FROM [Sales]

En este caso, como se omitió el parámetro *flags*, devolverá los miembros del nivel State Province que pertenezcan a Canadá. El mismo resultado dará si en *flags* seteamos,

SELF. Si en *flags* seteamos AFTER, hará un drill down al nivel inferior y devolverá los miembros, por ejemplo:

SELECT {[Measures].[Unit Sales]} ON COLUMNS,
(DESCENDANTS([customers].Canada, [State Province], AFTER)) ON ROWS
FROM [Sales]

En este caso, recibirá todos los Customers que pertenezcan a Canadá. Si utilizamos el flag *BEFORE,* devolverá sólo un miembro, *Canadá.* Con *BEFORE_AND_AFTER*, devolverá los miembros de ambos niveles, inferior y superior en la jerarquía.

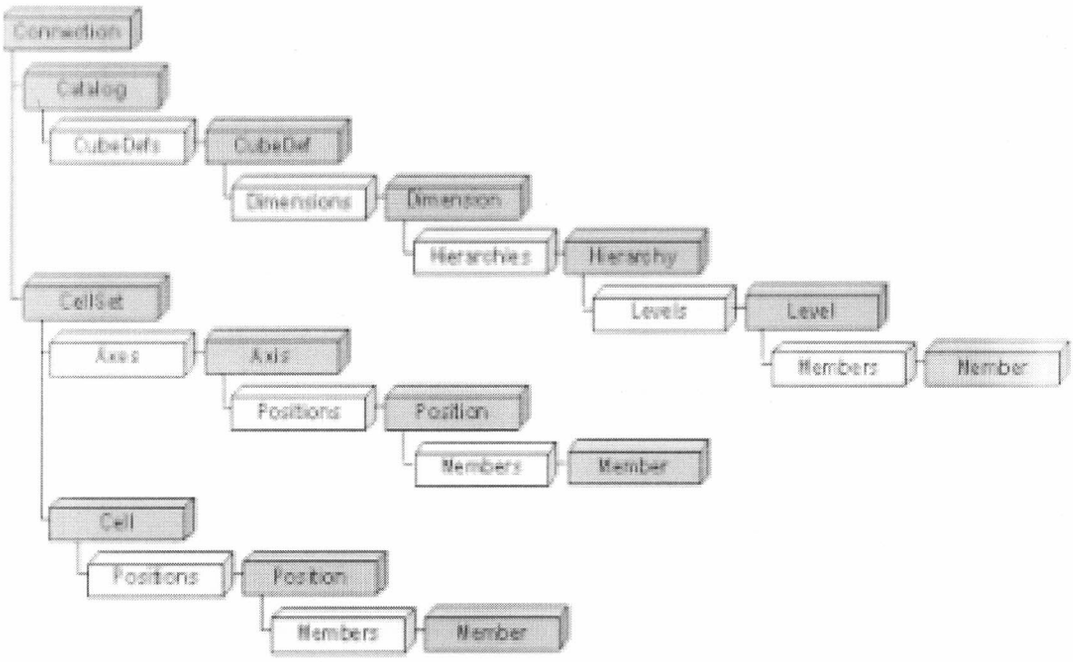
1.6 Extensiones ADO MD

La otra forma de obtener los datos del metadata y los datos del cubo es utilizando los objetos recomendados por Microsoft.

Los componentes *Microsoft Data Access Components (MDAC)* contienen más que los objetos básicos de ADO, incluyen extensiones para poder trabajar con bases de datos multidimensionales. Veremos el modelo de objetos y las funciones de las extensiones ADO MD. Estos objetos y funciones servirán para obtener información del metadata y de los datos de los cubos.

Objetos ADO MD

En el siguiente diagrama se muestran los objetos que pertenecen al modelo ADO MD.



Los objetos principales son Catalog y Cellset. El objeto *Catalog* contiene toda la información que describe a los cubos, estos nos permiten manipular Cubos, dimensiones, jerarquías, niveles y miembros.

- El objeto CubeDef, esta accesible desde la colección CubeDefs del objeto Catalog. Contiene la información detallada de un cubo. Este objeto representa a un cubo específico de OLAP.

- El objeto Dimensión, esta accesible desde la colección Dimensions del objeto CubeDef, contiene la información detallada de una dimensión. Este objeto representa a una dimensión específica.

- El objeto Hierarchy, esta accesible desde la colección Hierarchies. Hierarchy contiene la información detallada de una jerarquía. Este objeto representa a una jerarquía específica.

- El objeto Level, esta accesible desde la colección Levels. Level contiene la información detallada de un nivel de la jerarquía. Este objeto representa a un nivel específico.

- El objeto Member, esta accesible desde la colección Members del objeto Level. Member contiene la información detallada de una miembro de un nivel. Este objeto representa a un miembro específico.

El objeto *Cellset* contiene toda la información referida al resultado de la consulta multidimensional asociada, proveyendo un acceso a los datos como si fuera una colección o

un recordset. Permite traer información de los Axes y Cells y también de las Positions y Members.

- El objeto Axis, esta accesible desde la colección Axes, representa a un eje específico del *Cellset*. Contiene colecciones para acceder a los miembros resultado de una o más dimensiones.
- El objeto Cell, se puede acceder utilizando el método Item del objeto *CellSet*, Cell representa al dato en la intersección de los ejes (axis) actuales.
- El objeto Position, se accede a través de la colección Positions del objeto Cell. Representa al conjunto de uno o más miembros de diferentes dimensiones, el cual define un punto a lo largo de los ejes.

1.7 Aplicaciones OLAP

1.7.1 Componentes del lado del cliente

Al comienzo estudiamos que herramientas existían para acceder a los Análisis Services de Microsoft SQL Server 2000, encontramos el Decisión Cube que viene con Borland Delphi 5, este componente tiene una interfaz simplificada para acceder al Analysis Services, pero tiene muchas limitaciones, como la de no permitir la creación de cubos y el objeto de acceso a datos para hacer consultas no funciona correctamente con ADO.

Una de las ventajas de usar este componente, es que podemos brindar análisis OLAP sobre cualquier base de datos, no importa si tiene o no un motor de análisis, por ejemplo algunas bases de datos desktop como Interbase.

Las desventajas son bastante serias, pueden consumir una gran cantidad de recursos y gran tráfico en la red, ya que traen gran cantidad de datos desde el servidor al cliente.

En particular para el desarrollo del componente que servirá para la revisión de dimensiones y cubos, este modelo es inaceptable.

1.7.2 Componentes del lado del servidor

Un componente del lado del servidor a primera vista ya se presentaba como lo indicado, la ventaja principal es que evita enviar gran cantidad de datos al lado del cliente. En este caso todos los cálculos se hacen en el servidor y el cliente obtiene solamente el resultado.

Este esquema tiene una pequeña desventaja, vimos que fácilmente podemos utilizar OLE DB para las extensiones OLAP, puede suceder que otra base de datos, no provea el OLE DB para acceder a su OLAP o quizás lo provea pero con distinta funcionalidad,

Capitulo 2 Introducción a las Excepciones en cubos

2.1 Que son las Excepciones

Una base de datos multidimensional normalmente es representada como un conjunto de dimensiones y métricas, y la unidad de consulta es la celda. Una celda es un punto formado por la intersección de todas las dimensiones sobre un valor determinado de un nivel en la jerarquía de la dimensión. A este punto pueden asociarse una cantidad de métricas.

Las bases de datos multidimensionales, se enriquecieron en el tiempo, con el agregado de las jerarquías de agregación, permitiendo diferentes vistas a diferentes niveles de agregación. Dichas agregaciones pueden representarse como funciones roll-up entre niveles adyacentes dentro de una jerarquía, por otro lado para representar funciones roll-up entre niveles no-adyacentes, se puede utilizar la composición las funciones entre niveles adyacentes.

En muchos casos, estas funciones no pueden capturar el verdadero significado de la realidad, en particular cuando se quiere considerar en el análisis algún caso excepcional, ya que existirán casos en que queramos modificar sólo el contenido de alguna celda en particular, pero basados en algún criterio complejo, para estos casos es que se propone la definición de excepciones.

Estos casos de excepción pueden aparecer debido a que hay datos que no están accesibles, por políticas de la empresa o por incertidumbre en alguna condición al definir el Data Warehouse. En muchos de estos casos la composición de funciones roll-up no alcanzan para representarlos, por otro lado, la información en un Data Warehouse no siempre refleja de manera incuestionable la realidad de la empresa. A veces los datos pueden considerarse como una posible visión de una realidad que se intenta modelar, pero podría estar sujeta a revisión.

Creemos que es necesario un mecanismo de revisión que permita alterar el contenido de las instancias de las dimensiones y así lograr resultados más exactos y permitir tomar decisiones más acertadas. Estas revisiones también afectarán a los datos contenidos en los cubos definidos sobre las dimensiones afectadas por la revisión.

Las excepciones, muchas veces pueden servir para cambiar las reglas del negocio, para un determinado conjunto de miembros de una dimensión, pero también sirven para usarse como herramienta de análisis para estudiar posibles escenarios, para tratar de responder a la pregunta, *"Que pasaría si..."*

Permitiendo definir revisiones con sus reglas y aplicando los algoritmos de revisión, veremos cuales son las consecuencias de aplicar una excepción a un cubo multidimensional. Definir que es una excepción para esta tesis, tiene mucho que ver con el verdadero significado de la palabra excepción. Según el diccionario una *excepción* es algo extraordinario, fuera de lo normal, privilegiado y por extensión algo *excepcional* indica que

es algo que forma excepción de la regla común, que ocurre rara vez. Esta definición explica bastante bien que es lo que trataremos como excepción. Porque por ejemplo si definiéramos una revisión que me afectara a un 80% de las celdas de un cubo, no tendría sentido tratarlo como revisión y quizás debería ser modificada la definición del cubo o Data Mart.

Siguiendo la definición de excepción, al ser situaciones muy específicas, seguramente muchas de ellas no pudieron ser consideradas durante el proceso de construcción del Data Warehouse. Por eso queremos dar la facilidad de definir excepciones de forma interactiva, permitiendo definir reglas que logren suplantar las funciones roll-up compuestas y también permitan anular el efecto de las funciones roll-up para valores determinados.

Desarrollamos una aplicación que permite la definición de revisiones y también su ejecución, y en base a los resultados obtenidos analizaremos en que casos particulares los algoritmos presentados en [MV01] y [MV02] son viables para ser aplicados.

Otro beneficio que motiva la propuesta de permitir revisiones de datos, es que la posibilidad de definir excepciones, no sólo es comparable directamente con la modificación del Data Warehouse y el reprocesamiento del cubo, sino que también brinda mucha flexibilidad para los usuarios del Warehouse, ya que permite definir excepciones en cualquier momento, y no depender de expertos. Todo esto sin afectar al resto de los usuarios del Data Warehouse, ya que este no necesita ser modificado, ni reprocesado.

Veamos el siguiente ejemplo de una excepción: los lunes normalmente son considerados días laborables, supongamos que una falla en el suministro de energía ocurre el lunes 20/01/03, en una determinada sucursal, causando el cierre de la misma. En este caso la aseveración de que el lunes es un día laborable es una simple creencia y esta sujeto a duda. Por esto, dado que hay muchas jerarquías que son construidas sobre este tipo de conocimiento impreciso, y dado que los datos ya están almacenados en las instancias de las dimensiones, la posibilidad de definir una excepción, con capacidad para modificar el contenido de la instancia de una dimensión es muy importante. Estas revisiones también afectarán a las métricas.

A continuación presentaremos un ejemplo más detallado donde se evidencia la necesidad de definir excepciones.

Supongamos una compañía que otorga créditos. Esta compañía mantiene una base de datos con la información de todos los créditos organizados de la siguiente manera:

IdCredito: Identificador de crédito
IdCliente: Identificador de cliente
IdSucursal: Identificador de la sucursal donde se otorgó el crédito
Fecha: Fecha de aprobación

La siguiente tabla muestra el Fact Table de los créditos.

IdCredito	IdCliente	IdSucursal	Fecha	Monto
Cr1	C1	2	11/2002	20000
Cr2	C2	1	02/2002	5000
Cr3	C3	1	01/2003	120000
Cr4	C4	3	02/2003	58000

IdCliente, IdSucursal y Fecha son los niveles más bajos de las dimensiones, Cliente, Sucursales y Tiempo, mientras que Monto es la métrica e indica el monto del crédito

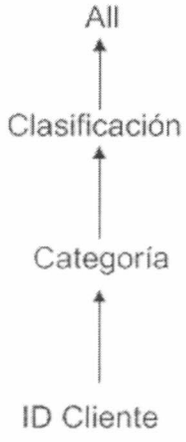


Figura A

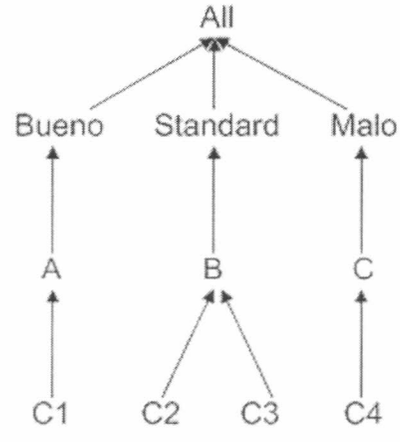


Figura B

En la figura A se muestra el esquema de la dimensión cliente y en la figura B una instancia de la misma. Cada nivel de la dimensión cliente a su vez tiene definidos atributos, el IdCliente tiene asociados dos atributos, Nombre e Ingresos. Cada miembro del nivel Categoría esta definido como un rango, este se implementó definiendo dos atributos, Inferior y Superior que indican el rango de Ingresos. Por último el nivel Clasificación también es un rango de las tasas de interés, también implementado como dos atributos Inferior y Superior.

En las próximas tablas veremos el contenido de las instancias de las dimensiones involucradas.

IdCliente	Nombre	Ingresos
C1	Jose Pérez	100000
C2	Pedro Juárez	30000
C3	Lucia García	25000
C4	Carlos Peña	10000

Tabla 1

Categoria	Inferior	Superior
A	50000	-1
B	25000	50000
C	0	25000

Tabla 2

Clasificacion	Inferior	Superior
Bueno	0,8	1
Standard	0,4	0,7
Malo	0	0,3

Tabla 3

Supongamos una consulta simple como: "*Listar el total de créditos acumulados por clasificación*". En ese caso los clientes *C2* y *C3* van a alimentar a la categoría *B*, acumulando según el fact table $125000, y la categoría *B* a su vez alimenta a la clasificación *Standard*, que recibe el mismo monto.

Ahora supongamos que por alguna política de la compañía, al cliente *C3* se lo quiere subir de Clasificación a *Bueno*. Esto podemos hacerlo definiendo una excepción como sigue:

Si el *IdClinete = C3 => Clasificacion = "Bueno"*

En este caso la clasificación *Bueno* ganará $120000, acumulando $140000 y la clasificación Standard pasará a tener $5000, perdiendo $120000.

Para dar solución a estas situaciones, lo que se hará es modificar la jerarquía como vemos en la figura C y esa es la jerarquía que deberá utilizar el algoritmo de agregación para resolver las consultas.

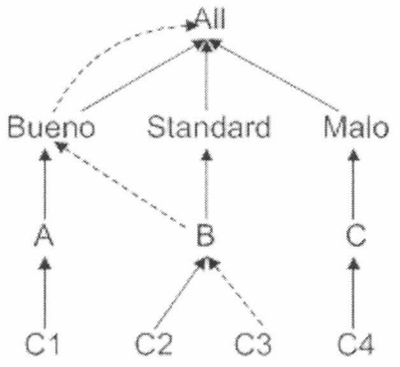


Figura C

En el ejemplo anterior vimos como la definición de excepciones dió solución a un caso, que de otra manera hubiera requerido regeneración de cubos, etc. Además si el cambio se quiere hacer para contestar la pregunta, *que pasaría si...*, veremos que definir una excepción es menos traumático que regenerar un cubo.

2.2 Síntesis

Debido a la naturaleza cambiante de los Data Warehouse, por las nuevas necesidades y requerimientos de los usuarios, sería muy importante dar la posibilidad de definir situaciones particulares una vez construido y procesado el Data Warehouse, estas situaciones se podrían salvar permitiendo manejar excepciones. Esta posibilidad permite cubrir casos excepcionales que de ninguna manera podían considerarse en la etapa de construcción del Data Warehouse, o que son situaciones fortuitas que ocurren en cualquier instancia del ciclo de vida del Data Warehouse.

Capítulo 3 Algoritmos para revisión de Cubos con Excepciones

3.1 Descripción de los algoritmos

Antes de comenzar a ver específicamente el desarrollo del componente de revisión, utilizando el lenguaje IRAH (Intensional Redefinition of Aggregation Hierarchies) definido en [MV01], veremos como son las estructuras y componentes básicos de las revisiones y las reglas, que son la base para definir excepciones sobre cubos.

Existen tres procesos básicos para el manejo de revisiones, estos son:

CreateRevision: con esta función se crean las revisiones asociadas a una dimensión, que sirven para la revisión de una dimensión o de un cubo.

CREATE REVISION revision_name
ON DIMENSION d
SET lh = ch
WHERE Cond1

Donde revision_name es el nombre identificador de la revisión.
La cláusula SET, asigna a un nivel (lh) de un path dentro de la jerarquía un valor (ch).
La cláusula WHERE determina que valores sobre un nivel son seleccionados, Cond1 es una condición con la forma descripta más adelante en este capítulo en *Gramática del parser*.

La tarea de crear revisiones se ejecuta del lado del cliente, su finalidad es definir las reglas de cada revisión, estas revisiones se almacenan en la base de datos para luego ser utilizadas por el proceso de revisión del lado del servidor.

ReviseDimension: ejecuta el proceso de revisión del cubo indicado.

REVISE DIMENSION d
WITH revision_name1
INTO revised_instance

Esta función sólo implementa la llamada al proceso central del servidor, pasándole como parámetro el nombre de la dimensión, el nombre de la revisión y donde dejar los resultados de la revisión.

ReviseCube: ejecuta el proceso de revisión del cubo indicado.

REVISE CUBE cube_name
WITH [revision_name1, TargetLevel1],[...], [revision_namek, TargetLevelk]
INTO revised_cube [[GAINED|LOST] VALUES ONLY]

La tarea de esta instrucción es la de ejecutar la revisión de un cubo, para eso se debe pasar como parámetros, el nombre del cubo a revisar, las dimensiones y para cada dimensión, un nombre de revisión, si es que se revisa y el TargetLevel elegido para cada dimensión. Además un nombre que será donde se guardarán los resultados de la revisión.

Las definiciones de las funciones en el lenguaje IRAH, son a modo explicativo, para ver cuales son los parámetros de una revisión, en la aplicación estas funciones están implementadas en métodos de objetos tales como Revision, ServerCube y ServerDimension.

A continuación daremos una explicación coloquial de cómo funciona el algoritmo de revisión de cubos, nos introducirá en el funcionamiento del algoritmo y las estructuras que maneja.

Al ejecutar la revisión de un cubo, el proceso cicla sobre las dimensiones a revisar, para cada una recorre los niveles de manera bottom-up y levanta los paths que vayan cumpliendo con las cláusulas WHERE definidas en las reglas para cada nivel.

Al finalizar esta primera etapa, selecciona cuales son las coordenadas de los niveles en los cuales hay aplicado un SET, dichas coordenadas se deben tener en cuenta para obtener las celdas del cubo afectadas por la revisión. Además se almacenan los niveles inferiores que provocaron la revisión, para luego realizar el recálculo de las celdas. Las celdas donde participen estas coordenadas sufrirán cambios en el valor de la métrica seleccionada, ya sea como pérdida (LOST) o como ganancia (GAINED).

Una vez determinadas las coordenadas para todas las dimensiones, se accede al cubo utilizando consultas MDX, para obtener la lista de celdas afectadas. Luego se recorre la lista de celdas afectadas y se calcula el nuevo valor de la métrica, ya sea LOST, GAINED o RESULT.

Una vez obtenidas las celdas recalculadas, se genera un cubo resultante conteniendo sólo las celdas modificadas y todas las dimensiones con los niveles a partir del Targetlevel para arriba.

3.2 Funcionalidad del componente

El desarrollo del componente se basó en el algoritmo del paper [MV01] que es el que permite la revisión de una dimensión en particular. Se definió el modelo de objetos para almacenar las reglas y las estructuras donde se almacenarían los Paths, Erules, Srules, etc. El detalle de estas estructuras las veremos en el punto 3.3 más adelante en este capítulo, cuando veamos el algoritmo de revisión de cubos.

Cuando se comenzó con el desarrollo de la interfaz, una de las primeras cosas que surgió fue la definición del lenguaje con que se escribirían las revisiones y las reglas. En

particular se tuvo que tratar de un modo especial las definiciones de las condiciones en las cláusulas WHERE, básicamente son expresiones que incluyen operadores de comparación entre strings y operadores lógicos como And, Or y Not, para interpretarlas se desarrolló un Parser cuya gramática vemos a continuación:

Gramática del parser

```
S -> K

K -> K AND L
K -> K OR L
K -> L

L -> E > F
L -> E < F
L -> E = F
L -> E >= F
L -> E <=  F
L -> E <> F

E -> NO E
E -> U

F -> NO F
F -> V

U -> ( K )
U -> FMD

V -> ( K )
V -> STR

FMD -> MemberDelCubo
STR -> String
```

Dentro de la gramática vemos el Identificador MemberDelCubo, este será el nombre de algún nivel de la dimensión o alguna propiedad de los niveles, en ambos casos deberá estar encerrado entre corchetes.

Cuando comenzamos a interactuar con el Análisis Server debimos determinar la mejor forma de implementar el acceso a los datos. Como vimos en el Capítulo 1 teníamos dos opciones para acceder a los datos multidimensionales, una era utilizando ADODB y su método OpenSchema y la otra utilizando los objetos recomendados por Microsoft Catalog y CellSet.

Al principio, como era lo recomendado, comenzamos utilizando el objeto Catalog para acceder al metadata de las dimensiones, pero probando sobre bases de datos reales, encontramos el problema de querer acceder a los miembros que satisfacían las condiciones

de las reglas y que los tiempos de respuesta eran muy pobres. Este fue el primer cambio grande, el problema con la estructura CATALOG es que la búsqueda es secuencial. La alternativa fue utilizar la librería ADODB y el objeto Connection con su método OpenSchema.

Utilizamos este método para obtener los miembros del último y el anteúltimo nivel, que son los de mayor cardinalidad. Este método nos permite, en una sola consulta, almacenar toda la información de los miembros en un Recordset de ADO y luego utilizando la propiedad Filter ir obteniendo los distintos subconjuntos que cumplan con determinada condición de una determinada regla. Queda como mejora determinar dinámicamente que niveles son susceptibles de ser tratados como voluminosos, y en ese caso utilizar el método OpenSchema en lugar de la estructura Catalog.

Las estructuras de datos fueron optimizadas para obtener la mejor performance posible. Debido a esto, las estructuras son dinámicas y en memoria, con paginación por demanda sobre el disco (esto dependerá de la cantidad de memoria RAM que tenga el servidor donde se ejecuta el componente).

La interfaz fue diseñada para mostrar la mayor información posible en una sola pantalla y tratando de simplificar la definición de las reglas. Debido a que la interfaz es a efectos de definir las revisiones y las reglas, se utilizó el modelo de objetos Catalog, que es el más ampliamente utilizado por Microsoft. (Ver Apéndice A)

Revisión de una dimensión

Como vemos en la figura 3.2.1, la parte de la aplicación encargada de realizar la revisión de una dimensión, esta compuesta por la interfaz y un componente que a partir de los datos de la instancia de la dimensión y de las revisiones que llegan como parámetro, dejará como resultado, un conjunto de Paths revisados, información de cómo fueron produciéndose los cambios en los Paths e información estadística de la ejecución.

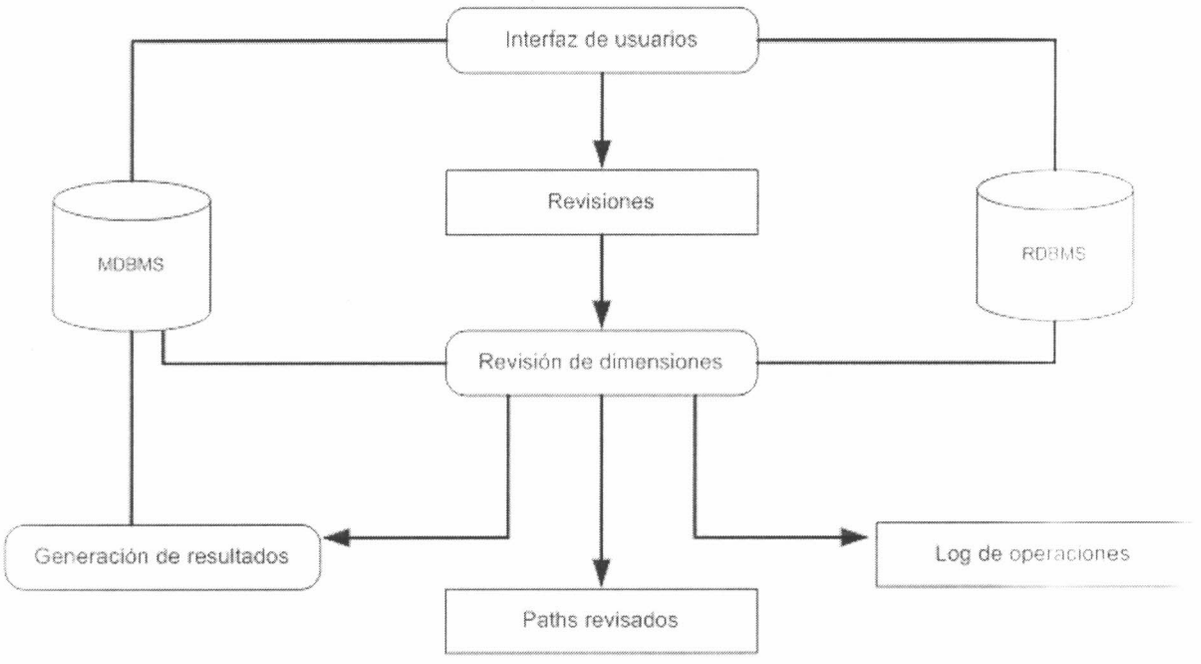


Figura 3.2 1

Una vez testeado el algoritmo de revisión de una dimensión, comenzamos con el desarrollo de los tres algoritmos del paper [MV02]. Al comenzar con el primero de los tres algoritmos se tomo la decisión de hacerlo en una nueva clase, ya que aunque la base era similar a lo desarrollado para la revisión de dimensiones, se preveía que iban a surgir muchas modificaciones y al tratarse este de un trabajo de investigación y análisis, se creyó mejor abrirlo, para que se mantengan como desarrollos independientes.

La nueva clase sufrió muchos cambios, comenzando por la interfaz, debido a que el algoritmo debía hacer la revisión de más de una dimensión, utilizando parámetros particulares para cada dimensión, así como ser la métrica sobre la cual se realiza el recálculo, los TargetLevels para cada dimensión, sean revisadas o no, y en el caso de las dimensiones que sean revisadas, la revisión que se va a aplicar.

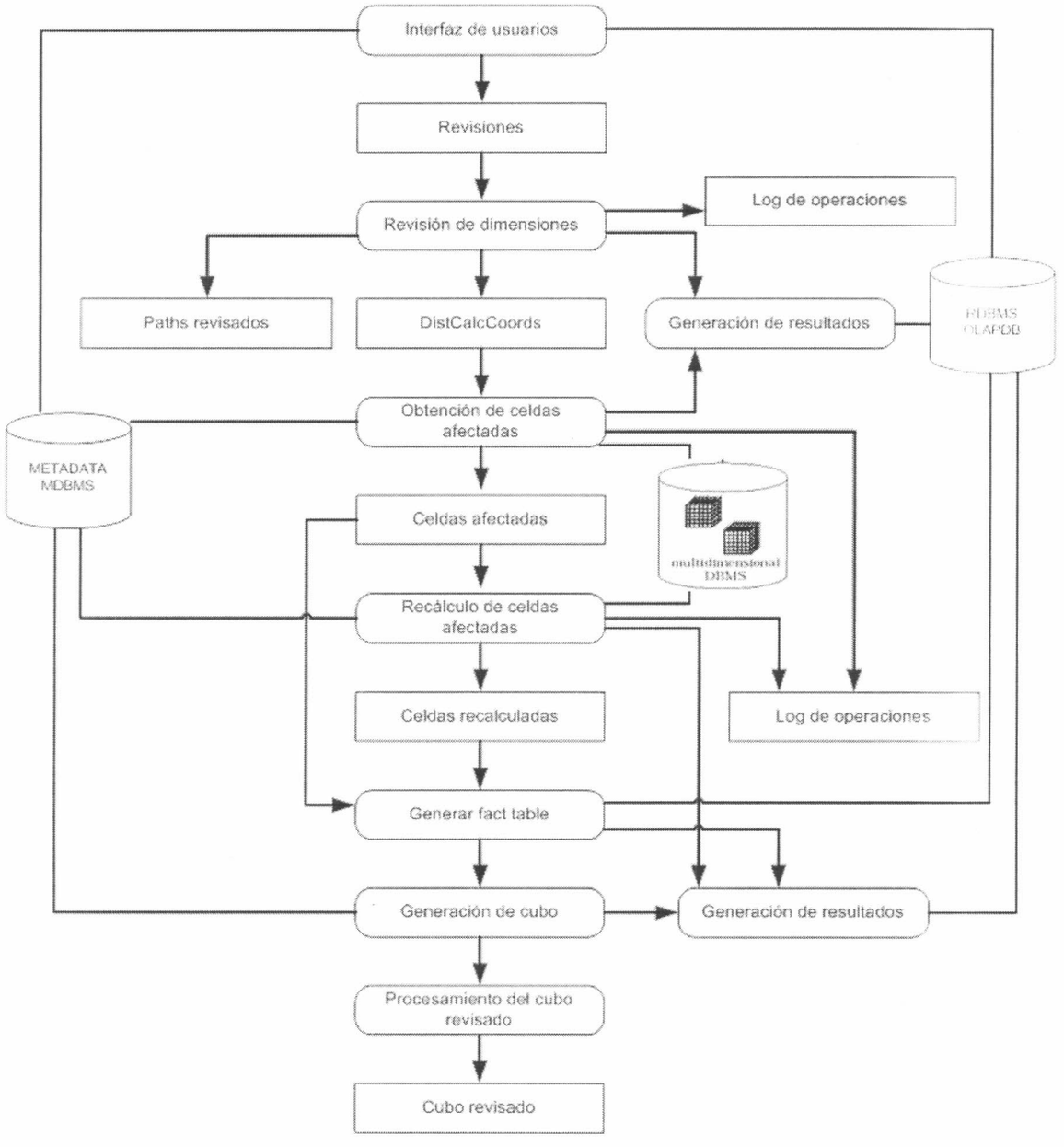


Figura 3.2 2

En el desarrollo de esta segunda parte se agregaron estructuras adicionales, así como funciones, grabación de datos de salida y de resultados para análisis (Figura 3.2 2). La entrada a la primera parte del algoritmo, son los datos del cubo a revisar, la métrica involucrada, y para cada dimensión el TargetLevel y la revisión, si corresponde.

El proceso recorre los niveles en los cuales existe una Cláusula WHERE aplicada y extrae los paths a revisar, modifica los niveles que correspondan de cada path y como salida dá un conjunto de paths revisados y una colección de DistCalcCoords para cada dimensión

revisada. DistCalcCoords es una estructura donde se indica para cada coordenada afectada por la revisión (valor en el SET de las reglas), cuales son las celdas origen de recálculo (Celdas Source) y el nivel de los nuevos valores. Esta estructura y las demás utilizadas, las veremos con más detalle más adelante en este capítulo.

El resultado de DistCalcCoords sirve como entrada para la segunda parte del algoritmo, donde para cada dimensión revisada se determina el conjunto de celdas del cubo afectadas por la revisión. Para determinar que celdas fueron afectadas por cada coordenada en DistCalcCoords, se hace una consulta multidimensional donde se fija el valor de los miembros del nivel indicado por TargetLevel de la dimensión revisada y se cruza con todos los valores de las demás dimensiones, para los TargetLevels indicados para cada dimensión. Este procedimiento se ejecuta para cada dimensión revisada. A continuación vemos un ejemplo de MDX para obtener las celdas afectadas.

```
WITH SET [DistCalc] AS 'Filter([Customer500_2].[City].Members, ([Customer500_2].CurrentMember.UniqueName =
"[Customer500_2].[All Customer1].[Country 0].[State Province 1].[City 3]") OR
([Customer500_2].CurrentMember.UniqueName = "[Customer500_2].[All Customer1].[Country 10].[State Province 20].[City
40]"))'
SELECT {[Measures].[Unit Sales]} ON AXIS(0) , [Product900].[Product Family].Members ON AXIS(1) , [DistCalc] ON
AXIS(2)
FROM [Sales500_900_0]
```

Ejemplo 3.2 3

En el ejemplo 3.2 3 las coordenadas del DistCalcCoords de la dimensión Cutomer son "City 3" y "City 40" en el nivel City. Y se cruza con la dimensión Product con TargetLevel "Product Family" y la métrica es "Unit Sales"

Las celdas afectadas son almacenadas por el algoritmo en recordsets. Existirán tantos recordsets de salida como dimensiones revisadas. Las filas de los recordsets contendrán los datos de las celdas afectadas, estos recordsets más la salida del primer algoritmo (DistCalcCoords), servirán como entrada para la tercer parte del algoritmo.

En esta etapa se recalcularán las celdas afectadas para obtener como salida los cubos Lost, Gained y Result del cubo revisado. Tanto los cubos analizados como los cubos de resultado, tienen un almacenamiento estilo MOLAP, ya que se preveía que otro modelo no iba a escalar de ninguna manera.

Lo que hará el algoritmo en esta etapa es, para cada celda afectada dentro de los resultset de salida del segundo algoritmo, basándose en los datos que provee la estructura DistCalcCoords para esa coordenada de esa dimensión, calculará el valor perdido y el ganado por esa celda debido a la revisión. Este procedimiento se hará para cada celda afectada (fila del recordset). Un MDX de ejemplo de recálculo sería el siguiente:

```
WITH SET [Dim1] AS 'Filter([Customer500_2].[Name].Members, ([Customer500_2].CurrentMember.Name = "Customer 7
7"))'
SELECT {[Measures].[Unit Sales]} ON AXIS(0) , NON EMPTY [Dim1] ON AXIS(1) , NON EMPTY [Product900].[Non-
Consumable].Children ON AXIS(2)
FROM [Sales500_900_0]
```

Ejemplo 3.2 4

Una vez recalculadas las celdas, el siguiente paso es generar el Fact. table base, sobre el cual se generará el nuevo cubo revisado. Por otro lado se generará también la estructura del cubo revisado en el Análisis Server. El nuevo cubo contendrá las mismas dimensiones que el cubo original, pero contendrá los niveles desde el TargetLevel para arriba, y contendrá cuatro métricas: la original a revisar, el valor perdido, el ganado y el resultado. Para la generación del nuevo cubo, debido a que se crea un cubo desde cero en el mismo Análisis Server, utilizaremos los DSO (Decision Support Objects) de Microsoft.

3.3 Estructuras de datos

Los algoritmos se basan en dos clases fundamentales. Para la revisión de dimensiones definimos la clase TServerDimension, la que maneja las siguientes estructuras para la ejecución de la revisión [MV01]:

***Cond**:* es un array bidimensional, con el índice de reglas en las filas y el índice de los niveles de la dimensión en las columnas. Cada celda no vacía de la matriz Cond[i,j] apunta a una fórmula para la regla i y el nivel j, esta fórmula puede ser para una cláusula Head o una cláusula Body. Las cláusulas Head son aquellas que definen el SET (*lo que se va a modificar*), mientras que las cláusulas Body, son aquellas que definen el WHERE (*los valores que causan que un path sea revisado*).

***Srules**:* es un array de arrays sobre los niveles de la dimensión, las posiciones en Srules[j,m] contienen punteros a las reglas donde la primer fórmula Body esta en el nivel j.

***Erules**:* es un array de arrays sobre los niveles de la dimensión, las posiciones en Erules[j,m] contienen punteros a las reglas donde la fórmula Head esta en el nivel j.

***Paths**:* es un array que contiene los paths recuperados de la instancia de la dimensión. Además asociado a cada valor c de cada nivel j de cada Path en Paths se guarda una variable que indica si ese valor fue modificado por la revisión o no.

Prules: es un array de arrays sobre los paths k en Paths, donde cada posición apunta a una regla i tal que el path k satisface la fórmula Body de la regla i.

***TargetPaths**:* es una estructura similar a Paths, donde se almacenan los Paths necesarios para ejecutar el proceso de Merge del algoritmo.

***Rpaths**:* es un array sobre las reglas, de la misma estructura de Paths. Cada celda Rpaths[i] apunta a un conjunto de paths en Paths, los cuales satisfacen la fórmula en el Body de la regla i.

La otra clase utilizada para la revisión de un cubo es TServerCube, las estructuras adicionales creadas para esta clase se detallan a continuación:

Body: es un array sobre las reglas, donde cada celda Body[i] contiene el máximo nivel j con una fórmula definida en ese nivel para la regla i.

Head: es un array sobre las reglas, donde cada celda Head[i] contiene el número de nivel j correspondiente a la fórmula Head de la regla i.

level_min: un array sobre Paths donde cada celda level_min[k] contiene el mínimo nivel con determinada condición en el Body para el path k.

Mcoord: es un array sobre los paths, donde cada celda Mcoord[k] apunta a un registro con la forma ((old: **Coordenada**, new: **Coordenada**, source: **Coordenada**, level: **Level**), old contiene el valor de la coordenada del path k para el nivel TargetLeve previo a la aplicación de la revisión, new contiene el valor de la coordenada del path k para el nivel TargetLevel luego de aplicar la revisión, y source y level contienen respectivamente la coordenada y el nivel de donde se aplicó efectivamente la primer regla de excepción de donde emanó el path k.

Arules: es un array sobre Paths, donde cada posición apunta a un conjunto de reglas i, tal que la regla i fue aplicada al path k.

Rcount: un array sobre los niveles y paths, donde cada celda Rcount[k,j] tiene la cantidad de reglas con el head en el nivel j, asociadas al path k.

Lpaths: un array sobre los niveles, cada posición Lpaths[j] apunta a otro array donde cada posición es un puntero a un path k en Paths, el cual esta asociado una regla con el Head en el nivel j.

DistCalcCoord: un array sobre las coordenadas del TargetLevel, cada posición DistCalcCoord[c] apunta a dos arrays, uno Gained y otro Lost, donde cada posición de cada array apunta a una dupla (source: Coordenada, level: Level).

3.4 Un ejemplo de revisión

A continuación veremos un caso práctico sobre el que aplicaremos una revisión sencilla, para ver todos los pasos por los que pasa el algoritmo.

Trabajaremos sobre la base de ejemplo Example2 (cuyo backup, relacional y multidimensional acompaña este trabajo) sobre las dimensiones Product, Customers y Time, del cubo Sales, utilizando la métrica [Unit Sales]. En las dimensiones que aparecen abajo, en negrita se muestran los Target Level para cada dimensión y debajo de cada nivel, su cardinalidad.

Dimensión Customers

All ——— Country ——————— State Province ——————— **City** ——————— Name
3 13 109 10281

Dimensión Product

All — Product Family — **Product Department** — Product Category — Product Subcategory — Brand Name — Product Name
3 25 56 102 512 1560

Dimensión Time

All ——— Year ——————— Quarter ——————— **Month**
2 8 24

La siguiente tabla muestra el contenido de una instancia del Fact Table base para generar el cubo.

product_id	time_id	customer_id	Promotion_id	store_id	store_sales	store_cost	unit_sales
931	606	5493	0	54	8,55	3,6765	25
931	570	7611	0	13	5,7	2,508	12
932	606	6429	0	44	8,55	3,42	24
932	570	2141	0	63	5,7	2,508	26
1553	606	181	0	24	8,55	4,104	23
1553	570	8966	0	63	11,4	4,902	28
1554	606	2102	501	54	11,4	3,99	35
1554	570	6429	0	75	11,4	4,902	61
588	570	1	0	10	8,55	2,9925	24
588	606	3	0	14	8,55	2,9925	18
592	606	9324	0	33	8,55	4,0185	8
592	570	4527	0	34	8,55	4,0185	56
1218	606	2690	0	82	8,55	4,0185	21
1218	570	3593	0	3	5,7	1,881	8
1529	570	9851	0	24	11,4	4,332	21
1529	606	7223	0	21	8,55	4,275	2
931	606	5493	0	54	8,55	3,6765	25

La revisión a aplicar será la siguiente

CREATE REVISION RevEj
ON DIMENSION Product
SET Set Product_Department = "Baked Goods"
WHERE Where [Product Subcategory] = "Jam"

REVISE CUBE Sales
WITH Product / RevEj
INTO SalesRev

Lo que hará esta revisión, es setear el valor del nivel Product_Department a lo que aparece en la cláusula SET, a aquellos paths que tengan un Product_Subcategory igual a lo que establece la cláusula WHERE. A continuación veremos los paths originales (PO) y su revisión (PR) que devuelve la implementación de la primera parte del algoritmo I del paper [MV02].

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[CDR].[CDR Apple Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[CDR].[CDR Apple Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Plato].[Plato Strawberry Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Plato].[Plato Strawberry Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Super].[Super Apple Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Super].[Super Apple Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Super].[Super Grape Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Super].[Super Grape Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Super].[Super Strawberry Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Super].[Super Strawberry Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[CDR].[CDR Grape Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[CDR].[CDR Grape Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[CDR].[CDR Strawberry Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[CDR].[CDR Strawberry Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Apple Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Apple Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Grape Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Grape Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Strawberry Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Landslide].[Landslide Strawberry Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Plato].[Plato Apple Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Plato].[Plato Apple Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[Plato].[Plato Grape Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[Plato].[Plato Grape Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Apple Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Apple Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Grape Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Grape Jam]

PO: [Product].[All Products].[Food].[Baking Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Strawberry Jam]
PR: [Product].[All Products].[Food].[Baked Goods].[Jams and Jellies].[Jam].[BBB Best].[BBB Best Strawberry Jam]

La segunda parte del algoritmo I, devolverá la siguiente estructura (DistCalcCoord) que será utilizada para saber que celdas se recalcularán, y cual será el nivel origen para hacer el recálculo.

DistCalcCoord[Baking Goods].Lost (Level = 2 [Subcategory], Source = "Jam")

DistCalcCoord[Baked Goods].Gained (Level = 2 [Subcategory], Source = "Jam")

Lo que se hará a partir de aquí es buscar todas las celdas que estén involucradas con alguno de los siguientes paths (Coordenadas) y el resultado serán las celdas que deberán ser recalculadas.

[All Products].[Food].[Baking Goods]
[All Products].[Food].[Baked Goods]

Para recalcular las celdas, se buscarán en el nivel indicado en DistCalcCoord, la coordenada indicada en el Source y eso se acumulará ya sea como Lost o como Gained.

Las siguientes tablas muestran como es una vista del cubo original, y las vistas de los cubos Lost, Gained y Resultado luego de la revisión.

Vista de cubo original

	Julio 1997	Agosto 1997
Baked Goods	85	49
Baking Goods	127	107

Vista del cubo Lost

	Julio 1997	Agosto 1997
Baked Goods	0	0
Baking Goods	127	107

Vista del cubo Gained

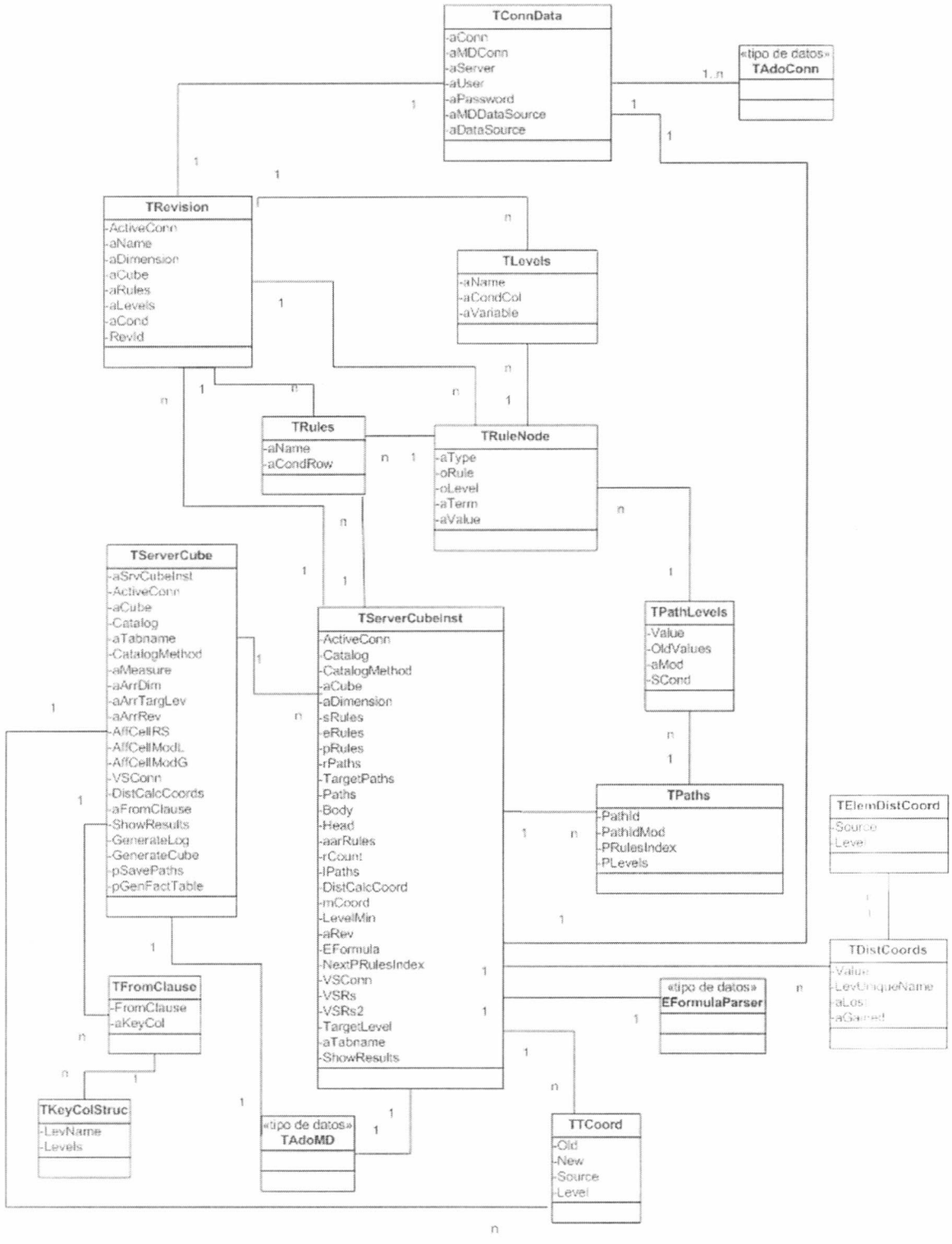
	Julio 1997	Agosto 1997
Baked Goods	127	107
Baking Goods	0	0

Vista del cubo Resultado

	Julio 1997	Agosto 1997
Baked Goods	212	156
Baking Goods	0	0

La aplicación del algoritmo dió como resultado cuatro Paths revisados y 144 celdas para recalcular. Que es un número muy inferior a la cantidad de celdas totales del cubo.

3.5 Modelo de objetos básico utilizado en la aplicación.


TConnData
-aConn
-aMDConn
-aServer
-aUser
-aPassword
-aMDDataSource
-aDataSource
«tipo de datos»
TAdoConn
TRevision
-ActiveConn
-aName
-aDimension
-aCube
-aRules
-aLevels
-aCond
-RevId
TLevels
-aName
-aCondCol
-aVariable
TRules
-aName
-aCondRow
TRuleNode
-aType
-oRule
-oLevel
-aTerm
-aValue
TServerCube
-aSrvCubeInst
-ActiveConn
-aCube
-Catalog
-aTabname
-CatalogMethod
-aMeasure
-aArrDim
-aArrTargLev
-aArrRev
-AffCellRS
-AffCellModL
-AffCellModG
-VSConn
-DistCalcCoords
-aFromClause
-ShowResults
-GenerateLog
-GenerateCube
-pSavePaths
-pGenFactTable
TServerCubeInst
-ActiveConn
-Catalog
-CatalogMethod
-aCube
-aDimension
-sRules
-eRules
-pRules
-rPaths
-TargetPaths
-Paths
-Body
-Head
-aarRules
-rCount
-lPaths
-DistCalcCoord
-mCoord
-LevelMin
-aRev
-EFormula
-NextPRulesIndex
-VSConn
-VSRs
-VSRs2
-TargetLevel
-aTabname
-ShowResults
TPathLevels
-Value
-OldValues
-aMod
-SCond
TPaths
-PathId
-PathIdMod
-PRulesIndex
-PLevels
TElemDistCoord
-Source
-Level
TDistCoords
-Value
-LevUniqueName
-aLos
-aGained
TFromClause
-FromClause
-aKeyCol
TKeyColStruc
-LevName
-Levels
«tipo de datos»
TAdoMD
«tipo de datos»
EFormulaParser
TTCoord
-Old
-New
-Source
-Level

3.6 Síntesis

En este capítulo presentamos brevemente lo que será implementado en el componente de revisión. Explicamos las estructuras teóricas a utilizar y vimos un ejemplo donde se detallaron los pasos por los que pasa el algoritmo durante la revisión. Los resultados son la parte más relevante de este trabajo, los analizaremos en el próximo capítulo. Para ver más detalles del desarrollo del componente de revisión remitirse a los Apéndice A y B.

Capítulo 4 Análisis empírico

Para realizar el análisis del comportamiento de los algoritmos, se tomarán varios casos modelo y evaluaremos distintos comportamientos sobre escenarios variados.

Modelos paramétricos de excepción

El algoritmo aplica revisiones sobre un cubo para soportar excepciones, por lo que necesitamos caracterizar las excepciones por medio de modelos paramétricos

Antes de definir los modelos paramétricos, daremos una definición de dos variables básicas para la realización de estas pruebas:

Celdas Afectadas: es un variable dependiente que surge de la aplicación del algoritmo. Se origina en los Paths que fueron revisados, en los valores indicados en el SET de las revisiones y además en la cardinalidad de los niveles seleccionados como TargetLevels en todas las dimensiones involucradas en el cubo a revisar. Sobre cada Celda Afectada se deberá recalcular el valor de la métrica elegida del cubo. El recálculo de estos valores se debe hacer celda por celda, ejecutando consultas MDX sobre el cubo, la ejecución de este tipos de queries insume un tiempo considerable, por lo que queremos demostrar que el tiempo de recálculo tiene relacion directa con el tiempo total que insume la revisión. Ademas queremos estudiar cómo se comporta el valor de *Celdas Afectadas* en relación a los modelos paramétricos que definiremos en esta sección.

Celdas Source: son aquellas que produjeron la pérdida o la ganancia de alguna de las Celdas Afectadas. De las Celdas Source se extraerá el valor que se acumulará con las demás Celdas Source relacionadas con una Celda Afectada, para producir el valor total de pérdida o ganancia. Las Celdas Source pertenecen a los niveles donde se cumplieron las condiciones de los WHERE, como para que ese path sea considerado por la revisión.

Para realizar este análisis, definiremos tres modelos paramétricos que caracterizan las excepciones y en base a ellos y a distintas revisiones definidas de modo tal de satisfacer dichos modelos, evaluaremos el comportamiento del algoritmo sobre instancias particulares de dimensiones y cubos. Recordamos como definimos una revisión, la cual puede contener una o más reglas (ver detalles en 3.1).

```
CREATE REVISION revision_name
ON DIMENSION d
SET lh1 = ch1
WHERE Cond1
...
SET lh2 = ch2
WHERE Cond2
...
```

Basándonos en esta definición de revisiones podemos ver que las reglas se definen sobre un nivel específico, por lo que, tanto el nivel que afecta el WHERE como el nivel que afecta el SET pueden verse como variables a tener en cuenta. Otra variable sería la cantidad de reglas definidas para una revisión de una dimensión.

Definiremos tres modelos paramétricos que denominaremos p-excepcion, l-excepcion y f-excepcion, estos intentarán reflejar como afectan al resultado final de la revisión, las variables provenientes de las reglas definidas en las revisiones y de las instancias particulares de las distintas dimensiones.

A continuación veremos las posibles variables involucradas en la definición de los modelos.

Al aplicar las reglas, dependiendo de su selectividad, el algoritmo nos dará distintos valores como resultado. Uno de estos será la cantidad total de **Paths revisados**, estos Paths son aquellos que fueron afectados por las condiciones en los WHERE. También la selectividad nos marca la cantidad de **Miembros seleccionados** en un nivel, y por último podemos considerar la cantidad de Paths seleccionados por una regla en cada nivel.

Respecto a las variables relacionadas con la instancia de la dimensión, consideramos la cantidad de **Paths totales** plausibles de ser revisados, esto se puede medir contando la cantidad de miembros en el último nivel. La cantidad total de **Miembros de cada nivel** y también la **Cantidad de niveles** involucrados en la jerarquía de la dimensión.

A continuación definiremos los tres tipos de modelos que reflejarán formas de ver la realidad, siempre relacionando las reglas con las instancias de las dimensiones.

p-excepciones

Este modelo refleja a grandes rasgos la relación entre las reglas y la instancia de la dimensión a revisar.

Por eso consideramos la *cantidad de paths revisados en la dimensión,* que es una variable relacionada con las reglas definidas en la revisión. Este valor indica la cantidad de Paths que cumplieron con las condiciones definidas en las reglas, surgidos como resultado de la ejecución de la primer parte del algoritmo.

Por otro lado tomamos una variable relacionada con la instancia de la dimensión, que es la cantidad total de Paths de la dimensión, este valor se obtiene contando la cantidad de miembros del nivel inferior de la jerarquía, ya que las jerarquías consideradas son completas.

Con estos valores definimos:

$$p\text{-}excepcion = \frac{cantidad\ de\ paths\ revisados\ en\ la\ \_dimension}{paths\ totales}$$

En este modelo se busca una medida global de como se comportan los algoritmos. Es una medida de la selectividad de las reglas. La cantidad de paths revisados tiene una relación importante con la forma en que fueron definidas las condiciones en las reglas. La finalidad de la definición de este modelo, es observar como se comporta en relación a la cantidad de Celdas Afectadas y a la Cantidad de Celdas Source, que son las que se deberán recalcular.

l-excepciones

Las p-excepciones reflejan la relación entre las condiciones de excepción y como éstas afectan a la instancia de una dimensión. Bajando más al detalle, veremos como es la distribución de los miembros dentro de las instancias de las dimensiones.

Considerando valores más intrínsecos a cada nivel dentro de la jerarquía, y siendo más específicos en cuanto a la selectividad de las reglas, no tomaremos los paths revisados, sino los miembros seleccionados por las reglas en cada nivel.

Tomamos los valores de *miembros seleccionados en el nivel L,* que es un valor relacionado con las condiciones de las reglas, ya que este valor esta relacionado con las condiciones en los WHERE, pero más específicamente relacionado con cada nivel donde se aplican y no tan generales como en las p-excepciones. Otro valor considerado será la *Cantidad de miembros del nivel L,* este valor es la cantidad de miembros que tiene cada nivel. Con estos dos valores definimos un modelo que nos indica cómo afecta la selectividad de las reglas, pero considerándolo nivel por nivel. Este modelo será un promedio de los valores de selectividad sobre los miembros para todos los niveles revisados.

$$l\text{-}excepcion = \frac{(\Sigma_{0..L}\ (Miembros\ seleccionados\ del\ nivel\ L\ /\ Cantidad\ de\ miembros\ del\ nivel\ L))}{Cant\ niveles\ revisados}$$

Este modelo paramétrico tiene en cuenta como están distribuidos los miembros en la jerarquía y también pone más énfasis en las características de los niveles donde están definidas las reglas.

f-excepciones

Definimos especialmente este modelo para estudiar su comportamiento donde existan más de una regla definida por revisión. Este modelo considera otra variable, que es la relación entre la cantidad de miembros de los niveles adyacentes. A esta nueva variable la llamaremos Fan In Out, que es un factor que relaciona la cantidad de miembros de un nivel con la del nivel superior, el cual nos da una idea de cómo se va abriendo el árbol, esto nivelará los pesos de los distintos niveles, a esta variable en adelante la denominaremos **FIO**.

Basándonos en el l-excepcion, definiremos al f-excepcion tomando los valores de *miembros seleccionados en el nivel L,* que es un valor relacionado con las condiciones de las reglas y la *Cantidad de miembros del nivel L ahora* afectado por el factor *FIO*, el cual nivelará los pesos de los distintos niveles donde se aplican las reglas. El *FIO* es un valor que se calcula por cada nivel a revisar dentro de la jerarquía.

$$f\text{-}excepcion = \frac{(\Sigma_{0..L}\ (Miembros\ seleccionados\ en\ el\ nivel\ L\ /\ (Miembros\ del\ nivel\ L\ /\ FIO)))}{Cantidad\ niveles\ revisados}$$

Definimos un modelo que muestra cómo afecta la selectividad de las reglas nivel por nivel, y haciendo una sumatoria de estos valores para todos los niveles revisados, y considerando la cantidad de niveles revisados, obtenemos un modelo que nos dará un promedio de selectividad de las de las reglas en relación a los miembros de cada nivel, pero a diferencia de l-excepcion, nivelando los pesos entre los niveles, para estudiar si la diferencia de cardinalidad afecta a los modelos..

Entorno de las pruebas

En esta sección definiremos las jerarquías de las dimensiones, los cubos modelo a utilizar y las revisiones que se aplicarán en las distintas pruebas a lo largo de este capítulo.

Dimensiones

Las dimensiones a considerar tienen las siguientes jerarquías:

La dimensión Dimension1 tiene la jerarquía

All ——— Level3 ——— Level2 ——— Level1 ——— Level0

La dimensión Dimension2 tiene la jerarquía

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0

Cubos

Presentaremos diez cubos que son la síntesis de todos los cubos sobre los que se realizaron las pruebas, el esquema de presentación de los cubos es:

Dimensiones:	Indica el nombre de las dimensiones del cubo y la cantidad de miembros de último nivel.
Métricas:	Indica el tipo de métrica que se considerará en las pruebas.
Storage:	Indica el modo de almacenamiento del cubo en el Análisis Server (MOLAP, ROLAP o HOLAP)
Tamaño:	Tamaño del cubo en Mb.

Celd. Materializadas:Cantidad de celdas materializadas.

N-Sparse: Indica el grado de dispersión de las celdas con contenido que hay en el cubo, por ejemplo en los cubos de los primeros grupos que probaremos, existen combinaciones de todos los miembros de una dimensión con todos los miembros de la otra dimensión en el Fact table, en ese caso serán 0-Sparse, los que veremos más adelante, serán 70-sparse, quiere decir que sólo el 30% de las combinaciones existen en el Fact table.

Cubo Sales_100_600_0:

Dimensiones:	Dimension1: 100 miembros
	Dimension2: 600 miembros
Métricas:	Sumatoria
Storage:	MOLAP
Tamaño:	0,88 Mb
Celd. Materializadas:	60000
N-Sparse:	0-Sparse

Cubo Sales_100_600_70:

Idem Anterior

N-Sparse:	70-Sparse

Cubo Sales_300_700_0:

Dimensiones:	Dimension1: 300 miembros
	Dimension2: 700 miembros
Métricas:	Sumatoria
Storage:	MOLAP
Tamaño:	1,13 Mb
Celd. Materializadas:	210000
N-Sparse:	0-Sparse

Cubo Sales_300_700_70:

Idem Anterior

N-Sparse:	70-Sparse

Cubo Sales_500_900_0:

Dimensiones:	Dimension1: 500 miembros
	Dimension2: 900 miembros
Métricas:	Sumatoria
Storage:	MOLAP
Tamaño:	2,17 Mb
Celd. Materializadas:	450000
N-Sparse:	0-Sparse

Cubo Sales_500_900_70:
Idem Anterior
N-Sparse: 70-Sparse

Cubo Sales_2000_900_0:
Dimensiones: Dimension1: 2000 miembros
Dimension2: 900 miembros
Métricas: Sumatoria
Storage: MOLAP
Tamaño: 9,40 Mb
Celd. Materializadas: 1800000
N-Sparse: 0-Sparse

Cubo Sales_2000_900_70:
Idem Anterior
N-Sparse: 70-Sparse

Cubo Sales5:
Dimensiones: Dimension1: 5000 miembros
Dimension2: 600 miembros
Métricas: Sumatoria
Storage: MOLAP
Tamaño: 0,4 Mb
Celd. Materializadas: 3000000
N-Sparse: 40-Sparse

Cubo SalesX9: (Cubo real, no generado)
Dimensiones: Dimension1: 10281 miembros
Dimension2: 1560 miembros
Dimension3: 24 miembros
Dimension4: 24 miembros
Dimension5: 6 miembros
Dimension6: 5 miembros
Dimension7: 2 miembros
Dimension8: 2 miembros
Métricas: Sumatoria
Storage: MOLAP
Tamaño: 5,23 Mb
Celd. Materializadas: -
N-Sparse: -

El contenido de los cubos fue generado por un algoritmo para realizar las pruebas sobre distintos modelos de datos.

Revisiones con una regla

Para el primer juego de pruebas definiremos cinco revisiones sobre la dimensión Dimension1 que tendrán el siguiente esquema:

Rev0010: Set [Level1] = C1 Where [Level0] > Cs0 (el valor Cs0, varía según el cubo)
Rev0011: Set [Level1] = C1 Where [Level0] > Cs1
Rev0012: Set [Level1] = C1 Where [Level0] > Cs2
Rev0013: Set [Level1] = C1 Where [Level0] > Cs3
Rev0014: Set [Level1] = C1 Where [Level0] > Cs4

Las variables C1 y Cs0 a Cs4 son sólo indicativas. Para cada instancia y cada regla, tomarán valores específicos que servirán para ir modelando las curvas. Estos valores fueron seleccionados para que las pruebas reflejen el comportamiento del algoritmo para distintos parámetros en los modelos. Una característica de estas revisiones es que todas contienen una única regla, únicamente variarán el rango que afectan en las distintas instancias de las dimensiones.

Revisiones con más de una regla

Para las pruebas con más de una regla por revisión definimos cinco revisiones sobre la dimensión Dimension1 que tendrán el siguiente esquema:

Rev0030:
r1: Set [Level3] = Ct1 Where [Level2] = SP21
r2: Set [Level2] = SP1 Where [Level1] >= C1 and [Level1] <= C3
r3: Set [Level1] = C40 Where [Level0] > Cs98

Rev0031:
r1: Set [Level3] = Ct1 Where [Level2] = SP22
r2: Set [Level2] = SP1 Where [Level1] >= C4 and [Level1] <= C12
r3: Set [Level1] = C40 Where [Level0] > Cs96

Rev0032:
r1: Set [Level3] = Ct1 Where [Level2] = SP23
r2: Set [Level2] = SP1 Where [Level1] >= C10 and [Level1] <= C18
r3: Set [Level1] = C40 Where [Level0] > Cs94

Rev0033:
r1: Set [Level3] = Ct1 Where [Level2] = SP24
r2: Set [Level2] = SP1 Where [Level1] = C20
r3: Set [Level1] = C40 Where [Level0] > Cs92

Rev0034:
r1: Set [Level3] = Ct1 Where [Level2] = SP25
r2: Set [Level2] = SP1 Where [Level1] = C20
r3: Set [Level1] = C40 Where [Level0] > Cs88

Revisiones para la Dimension2

Para las pruebas con más de una dimensión revisada, definimos cinco revisiones sobre la dimensión Dimension2 que tendrán el siguiente esquema:

RevProd30: r1: Set [Level3] = C1 Where [Level2] = SC1 or SC2
RevProd31: r1: Set [Level3] = C1 Where [Level2] = SC1 or SC2 or SC3
RevProd32: r1: Set [Level3] = C1 Where [Level2] = SC1 or SC2 or SC3 or SC4
RevProd33: r1: Set [Level3] = C1 Where [Level2] = SC1 or SC2 or SC3 or SC4 or SC5
RevProd34: r1: Set [Level3] = C1 Where [Level2] = SC1

Los valores de las variables C1 y SC1 – SC5, se variarán para que los valores de los modelos se sitúen dentro de los rangos a estudiar.

Revisiones para pruebas con Celdas Source

Definimos revisiones sobre Dimension1 para las pruebas de comportamiento en relación a las Celdas Source que tendrán el siguiente esquema:

Rev0001- Rev0017: r1: Set [Level1] = C1 Where [Level0] >= Cs1 And [Level0] <= Cs1

Identificadores de columnas de las tablas de resultados y sus unidades de medida

A continuación explicamos el significado de cada columna que aparecera en las tablas de las pruebas. No todas las columnas serán utilizadas para realizar los gráficos, algunas aparecen como información adicional.

TabName: Es un identificador con el que se almacenan los resultados obtenidos como ejecución del algoritmo de revisión. Estos datos podrán ser consultados desde la aplicación o directamente sobre la base de datos.

Dimensión: En el caso que haya varias dimensiones revisadas indica el nombre de la dimensión considerada. No aparece en el caso que se revise una sola dimensión.

PathsRevis.: Es la cantidad total de paths revisados en una dimensión.

Paths: Es la cantidad total de Paths de la dimensión con posibilidad de ser revisados.

Coords: Cantidad total de coordenadas que aparecen en el DistCalcCoords.

AffCells: Cantidad total de celdas afectadas por la revisión.

p-excep.: Valor que toma el modelo p-excepcion.

l-excep.: Valor que toma el modelo l-excepcion.

f-excep.: Valor que toma el modelo f-excepcion

T-MDX (1): Tiempo puro insumido para ejecutar todos los MDX's de recálculo de las celdas afectadas. Esta valor esta expresado en milisegundos.

T-Total: Tiempo total insumido en la revisión. Esta valor esta expresado en milisegundos.
Tsource: Cantidad total de Celdas Source consultadas efectivamente por el algoritmo, estas son las celdas source con valor en la métrica considerada, este valor aparece a modo indicativo, ya que no se utilizará en las pruebas.
Celdas/Seg: Cantidad de celdas recalculadas en 1 segundo.
Source: Es la cantidad de Celdas Source consideradas en el MDX. A diferencia de la columna TSource que definimos arriba, que indicaba la cantidad de celdas source consultadas, el valor de Source indica la cantidad elementos de filtro que se incluyeron en la consulta MDX, que es lo que realmente marca la diferencia en el tiempo de procesamiento de los MDX

Nota: Cada hilera de las tablas indica una revisión distinta sobre un cubo particular.

Resumen de las pruebas a realizar

A modo de introducción presentamos un resumen de las pruebas que se realizarán, testearemos el algoritmo sobre los diez cubos modelo definidos anteriormente. Para cada cubo ejecutaremos cinco o más revisiones, con una regla y con tres reglas por revisión, variando el p-excepcion de cada regla de 0 a 0,1. Estos valores arbitrarios de las cotas los consideramos apropiados ya que para el caso de las p-excepciones, un valor entre 0 y 0,1, representa una revisión con 10% de los paths revisados. El valor del p-excepcion fue tomado como punto de corte, siendo esta una evaluación empírica, ya que en cada instancia particular se podrían estudiar otros valores para p, pero se detectó que en este rango ya se va marcando una tendencia en el comportamiento de la curva. La cota para los valores del l-excepcion y f-excepcion se fijará a medida que vaya marcándose una tendencia.

Tambien se variarán en cada prueba la cantidad de miembos de los niveles de las dos dimensiones involucradas, para ver como afecta el tamaño de la dimensión a las variables seleccionadas, los valores de N-Sparse, la cantidad de dimensiones revisadas, la relación con las Celdas Source y también se hará un análisis de los tiempos insumidos en las revisiones.

Cada resumen de las pruebas.se presenta con un esquema similar al siguiente:

Título de la prueba: Título de la sección.

Y en un recuadro:

Objetivo Objetivo de las pruebas e índice del detalle.
Cubos: Cubos involucrados en la sección.
Sparse: Valor de N-Sparse de los cubos.
Dimensiones: Cantidad de Dimensiones de los cubos.
Dim. Revisada: Nombre de la dimensión revisada.
TargetLevel1: Target Level de la Dimensión revisada.

Revisiones: Revisiones aplicadas por el algoritmo.
Reglas x revisión: Cantidad de reglas por revisión .
TargetLevelN: TargetLevel de otras Dimensiones.
Variables: Variables utilizadas en las pruebas particulares.
Gráficos: Gráficos que aparecen en la sección.

A continuación presentamos el resumen de las pruebas.

Con una regla por revisión y una dimensión revisada

Objetivo	Comenzamos las pruebas con bajo volumen de información y estudiamos como se comportan los modelos definidos. El detalle de las pruebas lo veremos en la sección 4.1.
Cubos	Sales_100_600_0, Sales_300_700_0, Sales_500_900_0, Sales_2000_900_0
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014
Reglas x revisión	1
TargetLevel 2	Level5, nivel alto para producir baja cantidad de celdas afectadas.
Variables	Celdas afectadas, p-excepcion y l-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion y l-excepcion con las Celdas Afectadas.

Con una regla por revisión y mayor cantidad de celdas afectadas

Objetivo	Estudiar el comportamiento de los modelos definidos, con mayor cantidad de celdas afectadas. El detalle de las pruebas lo veremos en la sección 4.2.
Cubos	Sales_100_600_0, Sales_300_700_0, Sales_500_900_0, Sales_2000_900_0
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014
Reglas x revisión	1
TargetLevel 2	Level1, nivel bajo para producir alta cantidad de celdas afectadas.
Variables	Celdas afectadas, p-excepcion y l-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion y l-excepcion con las Celdas Afectadas.

Con una regla por revisión, con cubos 70-SPARSE

Cubos	Sales_100_600_70,Sales_300_700_70,Sales_500_900_70, Sales_2000_900_70
Sparse	70-Sparse
Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014
Reglas x revisión	1
TargetLevel 2	Level5, nivel alto para producir poca cantidad de celdas afectadas.
Variables	Celdas afectadas, p-excepcion y l-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion y l-excepcion con las Celdas Afectadas.

Con una regla por revisión, con cubos 70-SPARSE y mayor cantidad de celdas afectadas

Objetivo	Estudiar los modelos, con mayor cantidad de celdas afectadas, pero siguiendo con la misma dispersión de 70-Sparse, para ver si estas características afectan a las pruebas. El detalle de las pruebas lo veremos en la sección 4.4.
Cubos	Sales_100_600_70,Sales_300_700_70,Sales_500_900_70, Sales_2000_900_70
Sparse	70-Sparse
Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014
Reglas x revisión	1
TargetLevel 2	Level1, nivel bajo para producir alta cantidad de celdas afectadas.
Variables	Celdas afectadas, p-excepcion y l-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion y l-excepcion con las Celdas Afectadas.

Con más de una regla por revisión y una dimensión revisada

Objetivo	En esta prueba aumentar la cantidad de reglas definidas por revisión y mostrar como esto afecta a las curvas, en especial de l-excepcion y f-excepcion respecto de las celdas afectadas. El detalle de las pruebas lo veremos en la sección 4.5.
Cubos	Sales_100_600_0, Sales_300_700_0, Sales_500_900_0, Sales_2000_900_0
Sparse	0-Sparse

Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0030-Rev0034
Reglas x revisión	3
TargetLevel 2	Level5, nivel alto para producir poca cantidad de celdas afectadas.
Variables	Celdas afectadas, p-excepcion, l-excepcion, ademas utilizamos otro modelo paramétrico que es f-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion, l-excepcion y f-excepcion con las Celdas Afectadas.

Con más de una regla por revisión, una dimensión revisada y mayor cantidad de celdas afectadas.

Objetivo	Mostrar como afecta el aumento de la cantidad de reglas definidas por revisión a las curvas, en especial de l-excepcion y f-excepcion respecto de las celdas afectadas. Considerando mayor cantidad de celdas afectadas que en la sección anterior. El detalle de las pruebas lo veremos en la sección 4.6.
Cubos	Sales_100_600_0, Sales_300_700_0, Sales_500_900_0, Sales_2000_900_0
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0030-Rev0034
Reglas x revisión	3
TargetLevel 2	Level1, nivel bajo para producir gran cantidad de celdas afectadas, pero sin salirnos de las cotas.
Variables	Celdas afectadas, p-excepcion, l-excepcion. y f-excepcion
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion, l-excepcion y f-excepcion con las Celdas Afectadas.

Con más de una dimensión revisada

Objetivo	Mostrar como se comporta la curva producida en la revisión de ambas dimensiones juntas, en relación a las curvas producidas en las mismas condiciones pero por revisiones independientes de cada dimensión. El detalle de las pruebas lo veremos en la sección 4.7.
Cubos	Sales_100_600_0, Sales_2000_900_0. Los cubos intermedios se probaran de creerlo conveniente, ya que suponemos que las curvas se comportaran de manera similar a la revision independiente de las dimensiones, esto se debe a que el algoritmo las revisa a las dimensiones una a una..

Sparse	0-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014.
Reglas x revisión	1
Dim. Revisada 2	Dimension2
TargetLevel 2	Level3
Revisiones	RevProd30- RevProd34
Reglas x revisión	1
Variables	Celdas afectadas, p-excepcion y l-excepcion. Además aquí aparecen dos variables que relacionan los valores de los modelos para la revisión de cada dimensión, **p'-excepcion** es un factor que considera la relación entre las p-excepcion de cada dimensión y **l'-excepcion** de manera similar con las l-excepcion.
Gráficos	Presentamos gráficos que relacionan los valores p-excepcion, l-excepcion, p'-excepcion, l'-excepcion con las Celdas Afectadas.

Análisis de tiempos

En esta sección haremos pruebas para ver como se comporta la medida del Tiempo en relación a la Cantidad de Celdas Afectadas y Celdas Source. Y estimaremos el comportamiento del algoritmo en otras condiciones de entorno.

Con una regla por revisión y una dimensión revisada

Objetivo	Mostrar como se comportan los tiempos de recálculo y el tiempo total de la revisión en relación a las celdas afectadas por la revisión. Y también la relación entre las dos curvas. El detalle de las pruebas lo veremos en la sección 4.8.1
Cubos	Sales_100_600_0, Sales_300_700_0, Sales_500_900_0, Sales_2000_900_0. Esta prueba es la misma que la realizada en la sección 4.1, pero tendremos en cuenta las variables que se refieren al tiempo y no los modelos p-excepcion, l-excepcion y f-excepcion.
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014.
Reglas x revision	1

TargetLevel 2	Level5, nivel alto para producir poca cantidad de celdas afectadas.
Variables	Se tomarán valores de Tiempo puro de ejecución de las consultas MDX (T-MDX (1)), que se usan para recalcular las celdas afectadas por la revisión. Y también el tiempo total insumido (T-Total) por la revisión.
Gráficos	Presentamos gráficos que relacionan los valores de las celdas afectadas con T-MDX (1) y con T-Total.

Con una regla por revisión, una dimensión revisada y mayor cantidad de celdas afectadas

Objetivo	Mostrar como se comportan los tiempos (T-MDX (1)) y (T-Total), en relación a mayor cantidad de celdas afectadas por la revisión. Y también la relación entre las dos curvas. El detalle de las pruebas lo veremos en la sección 4.8.2
Cubos	Sales_100_600_0,Sales_300_700_0,Sales_500_900_0, Sales_2000_900_0. Esta prueba es la misma que la realizada en la sección 4.2, pero tendremos en cuenta las variables que se refieren al tiempo y no los modelos p-excepcion, l-excepcion y f-excepcion.
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014.
Reglas x revisión	1
TargetLevel 2	Level1, nivel bajo para producir gran cantidad de celdas afectadas.
Variables	Tomamos valores de Tiempo puro de ejecución de las consultas MDX (T-MDX (1)) y también el tiempo total insumido (T-Total) por la revisión.
Gráficos	Presentamos gráficos que relacionan los valores de las celdas afectadas con T-MDX (1) y con T-Total.

Prueba con un cubo con ocho dimensiones

Objetivo	El objetivo de estas pruebas es comprobar el comportamiento del tiempo cuando existen N dimensiones, la prueba será muy pequeña ya que estimamos que esto no afecta a los tiempos. El detalle de las pruebas lo veremos en la sección 4.8.3
Cubos	SalesX9.
Sparse	-
Dimensiones	8
Dim. Revisada 1	Dimension1

TargetLevel 1	Level3
Revisiones	Rev0002
Reglas x revisión	1
TargetLevel N	Level5, Level2, Level3, Level0, Level0, Level0, Level0
Variables	Celdas Afectadas, Tiempo-MDX y Tiempo Total.
Gráficos	No

Análisis del tiempo de procesamiento de las celdas

Objetivo	El objetivo de estas pruebas es estimar a grandes rasgos la viabilidad de ejecutar una revisión en términos de tiempos insumidos y dar una idea de si la misma podrá ser ejecutada de manera on-line u off-line. También servirá para estimar la potencia que deberá tener el hardware donde se ejecute la revisión. Se utilizarán como base los resultados de tiempos recopilados de todas las pruebas anteriores de tiempos realizadas en 4.8.1 y 4.8.2. Estas son pruebas sobre ocho cubos, con distinto TargetLevel sobre la Dimension2 para variar la cantidad de celdas afectadas. El detalle de las pruebas lo veremos en la sección 4.8.4
Cubos	Sales_100_600_0,Sales_300_700_0,Sales_500_900_0, Sales_2000_900_0.
Sparse	0-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014.
Reglas x revisión	1
TargetLevel 2	Level5 y Level1, cuatros pruebas cada uno..
Variables	Definimos las siguientes variables, **Celdas materializadas** como la cantidad de celdas materializadas del cubo. **Celdas por segundo** como la cantidad de celdas que se pueden procesar por segundo. **Cant. Procesadores** como la cantidad de procesadores necesarios para mantener un tiempo constante de procesamiento por celda a medida que aumenta el volumen del cubo. **Celdas a procesar** es la cantidad de celdas a procesar. **Tiempo (Seg)** es el tiempo insumido en el procesamiento de n celdas.

Gráficos	En esta sección presentamos tres gráficos, uno que relaciona la cantidad de Celdas materializadas del cubo con la cantidad de celdas que se pueden procesar en un segundo. Veremos que a medida que aumenta el tamaño del cubo disminuirá la cantidad de celdas que se pueden procesar por segundo. El segundo gráfico mostrará la relación entre el tamaño del cubo, con la cantidad de procesadores necesarios para mantener un tiempo de procesamiento por celda constante. Este gráfico muestra como debería ir aumentando la potencia del hardware a medida que crece el volumen del cubo. Por último se muestra un gráfico que relaciona la cantidad de celdas a procesar, respecto del tiempo, considerando tiempos constantes de procesamiento por celda, en este gráficos veremos que el crecimiento es lineal, lo que indica que el algoritmo es predecible y no evidencia picos.

Análisis del tiempo en relación a las Celdas Source

Se realizarán pruebas de tiempos y de comportamiento de las p-excepciones y l-excepciones en relacion con las Celdas Source consideradas para realizar los filtros de las consultas MDX ejecutadas para el recálculo de las celdas afectadas. Ademas realizaremos estudios de tiempos insumidos en la revisión.

Análisis de tiempos

Objetivo	Mostrar el comportamiento del tiempo de recálculo de las celdas (Tiempo MDX) en relación con distintos valores de Celdas Source (Source). El detalle de las pruebas lo veremos en la sección 4.9.1
Cubos	Sales5000.
Sparse	40-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0001-Rev0017.
Reglas x revisión	1
TargetLevel 2	Level5
Variables	Tiempo-MDX, Source, y además **PSource** que es un coeficiente que relaciona la cantidad de Celdas Source consideradas por cada Celda Afectada.
Gráficos	Presentamos dos gráficos, el primero de ellos mostrando como se comporta el tiempo de recálculo (Tiempo MDX) respecto de las Celdas Source de filtro (Source). El segundo muestra como se comporta el tiempo (Tiempo MDX) respecto a la cantidad de Celdas Source por Celda Afectada (PSource).

Análisis respecto de las p-excepciones y l-excepciones

Objetivo	Mostrar el comportamiento de las Celdas Source respecto de los modelos definidos p-excepcion y l-excepcion, se omitió la prueba con f-excepcion ya que se utilizaron revisiones con una sola regla. El detalle de las pruebas lo veremos en la sección 4.9.2
Cubos	Sales5000.
Sparse	40-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0001-Rev0017.
Reglas x revisión	1
TargetLevel 2	Level5
Variables	PSources, p-excepcion y l-excepcion
Gráficos	Presentamos dos gráficos que relacionan los valores de PSources con los modelos p-excepcion y l-excepcion.

Pruebas relacionando las Celdas Source por Celda afectada

Objetivo	Mostrar como se comporta el tiempo de recálculo de las celdas (Tiempo MDX) considerando distintos valores de CeldasT, que es el producto entre las Celdas Afectadas y las Celdas Source filtro de los MDX. El detalle de las pruebas lo veremos en la sección 4.9.3
Cubos	Sales100_600_0, Sales2000_900_0.
Sparse	40-Sparse
Dimensiones	2
Dim. Revisada 1	Dimension1
TargetLevel 1	Level1
Revisiones	Rev0010-Rev0014
Reglas x revisión	1
TargetLevel 2	Level5 y Level1
Variables	Tiempo-MDX, y definimos **CeldasT** como el resultado del producto entre la cantidad de Celdas Afectadas por el valor de Source, que es el indicador de las Celdas Source por las que se filtra el MDX de recálculo.
Gráficos:	Presentamos cuatro gráficos para ver el comportamiento de las curvas del tiempo de recálculo (Tiempo MDX) respecto de las CeldasT para distintos volúmenes en los cubos, distinta cantidad de Celdas Afectadas y distinta cantidad de Celdas Source.

Esquema de las pruebas

Todas las pruebas se asimilan al siguiente esquema:

- Título de la prueba. Este aparecerá con el siguiente formato:

  [Nombre 1ra Dimensión]-[Miembros último nivel]-[TargetLevel] [Nombre 2da Dimensión]- [Miembros último nivel]- [TargetLevel] [Opciones]

  Donde en *[Opciones]* puede aparecer el modelo de Sparse, que es una medida de la distribución de celdas no vacías en el cubo, respecto de las posibles celdas, o también la cantidad de dimensiones a revisar, o reglas involucradas en la revisión.

- Objetivo y parámetros de la prueba.
- Información de las dimensiones. Aquí se mostrará un gráfico de la jerarquía de niveles de las dimensiones, en estos se marcará la cardinalidad debajo de cada nivel, y en negrita el TargetLevel para esa prueba.
- Tabname's utilizados para las distintas pruebas. Estos servirán para ver la información en la base de datos y en la aplicación.
- Nombre del cubo dentro del Análisis Server.
- Rango de revisiones utilizadas.
- Tabla con valores que devuelve el algoritmo. Se representarán en los gráficos.
- Gráficos con curvas para analizar.
- Conclusión, si corresponde.

4.1 Pruebas con una regla por revisión y una dimensión revisada

En esta sección aplicamos revisiones sobre la primera de las dimensiones, que es Dimension1 utilizando como TargetLevel el nivel Level1, y como TargetLevel de Dimension2 tomamos Level5, Dimension2 no será revisada.

Sobre el eje de las Y aparece la cantidad de *Celdas Afectadas*. En todos los casos de esta sección aparecerán dos gráficos, uno con el eje de las X mostrando el p-excepcion y otro mostrando el l-excepcion.

Dimension1-100-Level1 Dimension2-600- Level5 0-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100, y definimos reglas que vayan tomando distintos valores de *Paths revisados* entre el 0% y el 10%, para que p-excepcion vaya variando entre 0 y 0,1 aproximadamente. El TargetLevel de la Dimension2, es uno de nivel superior con baja cardinalidad, para obtener valores bajos de celdas afectadas.

Lo consideramos así, porque la cardinalidad de los niveles de los TargetLevels da una medida de las celdas que se afectarán, ya que una celda está determinada por el cruce de todas las dimensiones en un valor determinado para cada nivel de la jerarquía de cada dimensión y las celdas afectadas se obtienen de cruzar la coordenada afectada en la dimensión revisada con todos los miembros de las demás dimensiones.

Dimension1

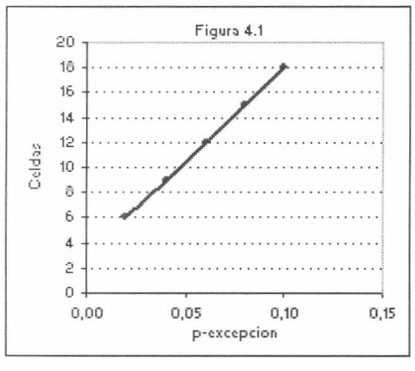
All	Level3	Level2	**Level1**	Level0
	13	26	51	100

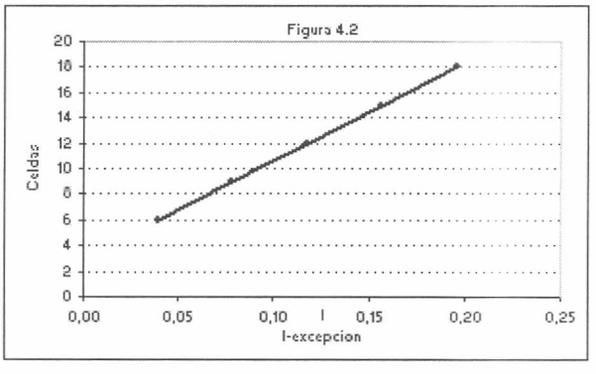
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	199	600

TabName's: Cubo0001_1 - Cubo0001_2 - Cubo0001_3 - Cubo0001_4 - Cubo0001_5
Cubo: Sales100_600_0
Revisiones: Rev0010 - Rev0014

Tabéame	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0001_1	2	100	2	6	0,020	0,020	30,00	1562,00	92	200,00
Cubo0001_2	4	100	3	9	0,040	0,040	40,00	1982,00	184	225,00
Cubo0001_3	6	100	4	12	0,060	0,060	50,00	1021,00	276	240,00
Cubo0001_4	8	100	5	15	0,080	0,080	50,00	1523,00	368	300,00
Cubo0001_5	10	100	6	18	0,100	0,100	70,00	1372,00	460	257,14





Dimension1-300-Level1 Dimension2-700- Level5 0-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 300, y continuamos utilizando reglas que tomen distintos valores de *Paths revisados* entre el 0% y el 10%. El TargetLevel para la Dimension2 es Level5. Aumentamos la cantidad de Paths totales para estudiar como se comportan los modelos.

Dimension1

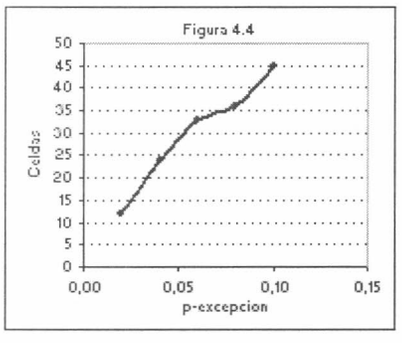
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
38 76 151 300

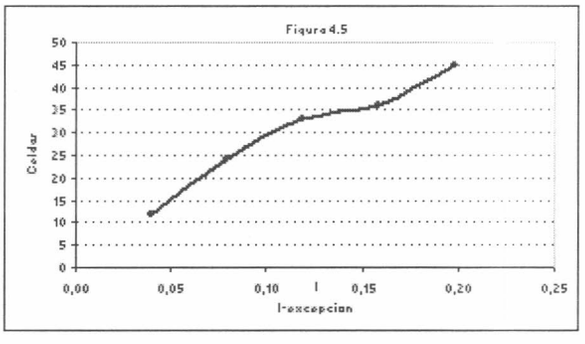
Dimension2

All ——— **Level5** ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0
3 23 55 102 231 700

TabName's: Cubo0002_1 - Cubo0002_2 - Cubo0002_3 - Cubo0002_4 - Cubo0002_5
Cubo: Sales300_700_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0002_1	6	300	4	12	0,020	0,020	100,00	1422,00	276	120,00
Cubo0002_2	12	300	8	24	0,040	0,040	220,00	2133,00	552	109,09
Cubo0002_3	18	300	11	33	0,060	0,060	270,00	1993,00	828	122,22
Cubo0002_4	24	300	12	36	0,080	0,080	360,00	1872,00	1012	100,00
Cubo0002_5	30	300	15	45	0,100	0,100	480,00	2293,00	1288	93,75





Dimension1-500-Level1 Dimension2-900- Level5 0-Sparse

Nuevamente aumentamos la cardinalidad en los niveles de las dimensiones conservando las demás características.

Dimension1

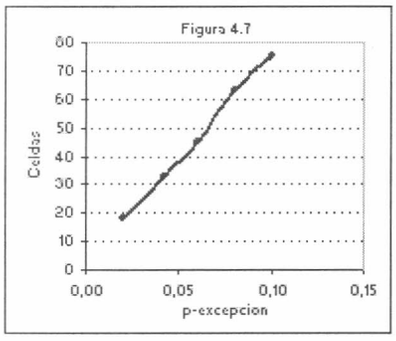
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
63 126 251 500

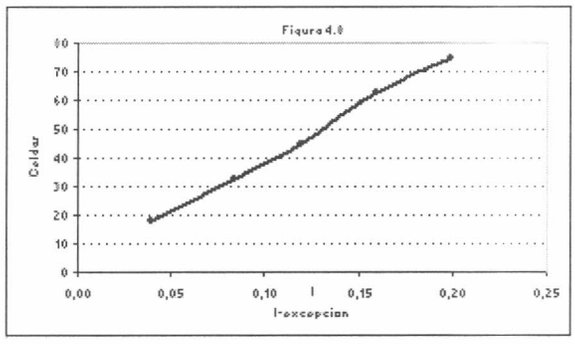
Dimension2

All ——— **Level5** ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0
3 23 55 102 295 900

TabName's: Cubo0003_1 - Cubo0003_2 - Cubo0003_3 - Cubo0003_4 - Cubo0003_5
Cubo: Sales500_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0003_1	10	500	6	18	0,020	0,020	211,00	1823,00	460	85,31
Cubo0003_2	21	500	11	33	0,042	0,042	561,00	2674,00	874	58,82
Cubo0003_3	30	500	15	45	0,060	0,060	711,00	3314,00	1288	63,29
Cubo0003_4	40	500	21	63	0,080	0,080	891,00	3014,00	1748	70,71
Cubo0003_5	50	500	25	75	0,100	0,100	1091,00	3955,00	2208	68,74





Dimension1-2000-Level1 Dimension2-900- Level5 0-Sparse

En la última prueba de esta sección, aumentamos la cardinalidad en los niveles de las dimensiones.

Dimension1

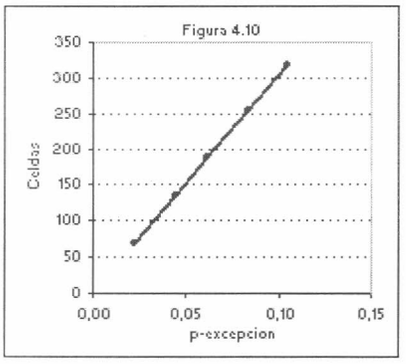
All	Level3	Level2	**Level1**	Level0
	251	501	1001	2000

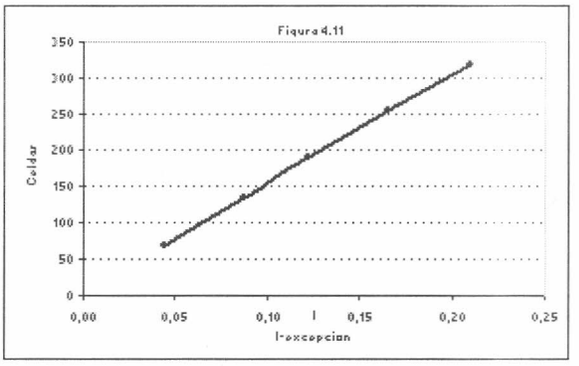
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	295	900

TabName's: Cubo0004_1 - Cubo0004_2 - Cubo0004_3 - Cubo0004_4 - Cubo0004_5
Cubo: Sales2000_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0004_1	44	2000	23	69	0,022	0,022	3275,00	6129,00	2024	21,07
Cubo0004_2	88	2000	45	135	0,044	0,044	6810,00	11817,00	4048	19,82
Cubo0004_3	122	2000	63	189	0,061	0,061	9253,00	15792,00	5612	20,43
Cubo0004_4	166	2000	85	255	0,083	0,083	12559,00	18567,00	7636	20,30
Cubo0004_5	210	2000	106	318	0,105	0,105	15994,00	23314,00	9614	19,88





En las cuatro pruebas anteriores vimos que a pesar de ir aumentando la cardinalidad de los niveles de las dimensiones involucradas, las curvas se comportan de manera similar, con algunas variaciones. Siempre con un crecimiento proporcional de las celdas afectadas a medida que crecen los los valores de los modelos p-excepion y l-excepcion.

Podemos concluir para este caso que el modelo p-excepcion tiene relación directa con las celdas afectadas, debido a que las reglas están aplicadas sobre un nivel bajo de la dimensión y que al existir una regla por revisión existe una única cláusula SET.

En este caso l-excepcion, se comporta de manera similar al modelo p-excepcion, ya que se aplican condiciones sobre un sólo nivel, este modelo mostrará mayor variación cuando se afecten más niveles con más de una regla por revisión.

4.2 Pruebas con una regla por revisión pero más celdas afectadas

Continuando con los mismos cubos, estudiamos como afecta a las curvas el incremento de la cantidad de celdas afectadas, por lo que bajamos el nivel del TargetLevel de la Dimensión2 a Level1, de esta manera producimos mayor cantidad de celdas a recalcular.

Dimension1-100-Level1 Dimension2-600- Level1 0-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100, y definimos reglas que tomen distintos valores de *Paths revisados* entre el 0% y el 10% aproximadamente. Bajamos el TargetLevel de la Dimension2 a Level1 para obtener más celdas afectadas.

Dimension1

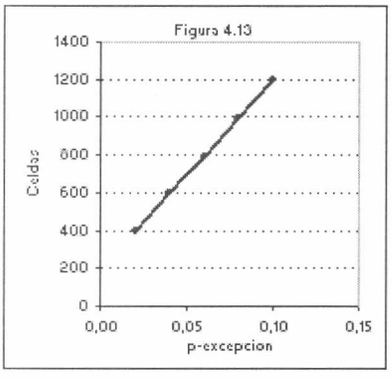
All	Level3	Level2	**Level1**	Level0
	13	26	51	100

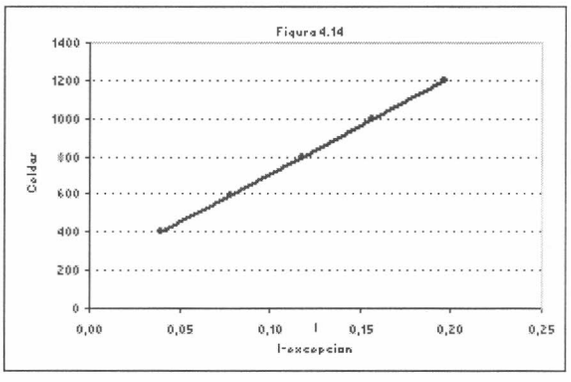
Dimension2

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 199 600

TabName's: Cubo0011_1 - Cubo0011_2 - Cubo0011_3 - Cubo0011_4 - Cubo0011_5
Cubo: Sales100_600_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0011_1	2	100	2	398	0,020	0,020	0,039	1302,00	4706,00	2400	305,68
Cubo0011_2	4	100	3	597	0,040	0,040	0,078	2534,00	18726,00	4800	235,60
Cubo0011_3	6	100	4	796	0,060	0,060	0,118	3766,00	8492,00	7200	211,36
Cubo0011_4	8	100	5	995	0,080	0,080	0,157	4165,00	12578,00	9600	238,90
Cubo0011_5	10	100	6	1194	0,100	0,100	0,196	5470,00	10596,00	12000	218,28





Dimension1-300-Level1 Dimension2-700- Level1 0-Sparse

Tomamos un valor de *Paths Totales* de 300 para la dimensión revisada y un TargetLevel con más cardinalidad para la Dimension2.

Dimension1

All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
38 76 151 300

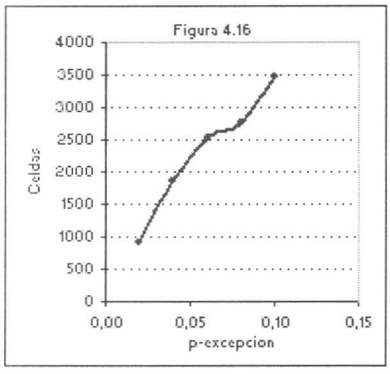
Dimension2

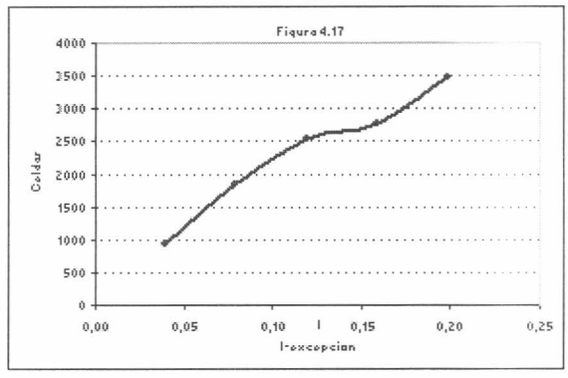
All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 231 700

TabName's: Cubo0012_1 - Cubo0012_2 - Cubo0012_3 - Cubo0012_4 - Cubo0012_5
Cubo: Sales300_700_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg

Cubo0012_1	6	300	4	924	0,020	0,020	6969,00	11837,00	8400	132,59
Cubo0012_2	12	300	8	1848	0,040	0,040	15210,00	22112,00	16800	121,50
Cubo0012_3	18	300	11	2541	0,060	0,060	22904,00	30624,00	25200	110,94
Cubo0012_4	24	300	12	2772	0,080	0,080	28452,00	37524,00	30800	97,43
Cubo0012_5	30	300	15	3465	0,100	0,100	34090,00	43854,00	39200	101,64





Dimension1-500-Level1 Dimension2-900- Level1 0-Sparse

Para Dimension1 tomamos un valor de *Paths Totales* de 500 y un TargetLevel con más cardinalidad para la Dimension2.

Dimension1

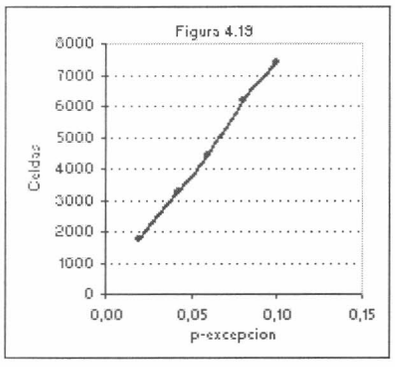
All	Level3	Level2	**Level1**	Level0
	63	126	251	500

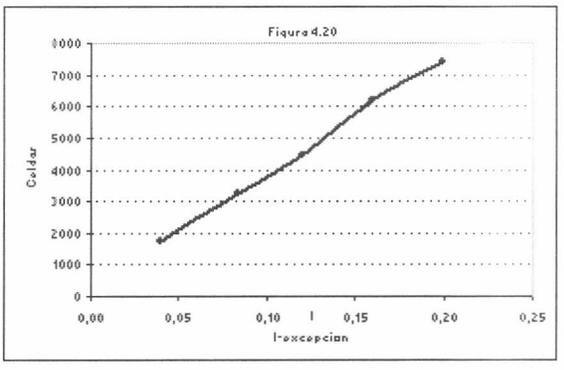
Dimension2

All	Level5	Level4	Level3	Level2	**Level1**	Level0
	3	23	55	102	295	900

TabName's: Cubo0013_1 - Cubo0013_2 - Cubo0013_3 - Cubo0013_4 - Cubo0013_5
Cubo: Sales500_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0013_1	10	500	6	1770	0,020	0,020	21339,00	26318,00	18000	82,95
Cubo0013_2	21	500	11	3245	0,042	0,042	45185,00	52065,00	34200	71,82
Cubo0013_3	30	500	15	4425	0,060	0,060	70042,00	81247,00	50400	63,18
Cubo0013_4	40	500	21	6195	0,080	0,080	95989,00	111681,00	68400	64,54
Cubo0013_5	50	500	25	7375	0,100	0,100	123048,00	164486,00	86400	59,94





Dimension1-2000-Level1 Dimension2-900- Level1 0-Sparse

Como última prueba de esta sección, probamos con más celdas afectadas y un total de paths de 2000.

Dimension1

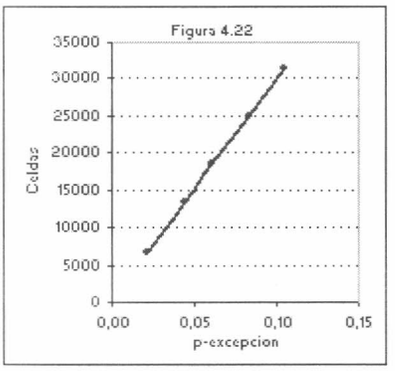
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
251 501 1001 2000

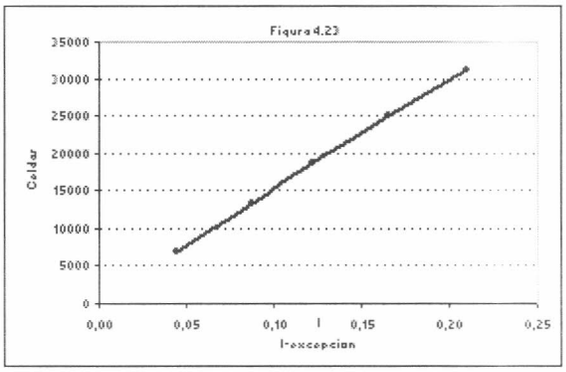
Dimension2

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 295 900

TabName's: Cubo0014_1 - Cubo0014_2 - Cubo0014_3 - Cubo0014_4 - Cubo0014_5
Cubo: Sales2000_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0014_1	44	2000	23	6785	0,022	0,022	355318,00	370694,00	79200	19,10
Cubo0014_2	88	2000	45	13275	0,044	0,044	703575,00	743400,00	156857	18,87
Cubo0014_3	122	2000	63	18585	0,061	0,061	999000,00	1101684,00	219600	18,60
Cubo0014_4	166	2000	85	25075	0,083	0,083	1328975,00	1404200,00	296286	18,87
Cubo0014_5	210	2000	106	31270	0,105	0,105	1657310,00	1751120,00	369486	18,87





En estos últimos cuatro casos de prueba, las curvas conservan el comportamiento de la sección anterior, tanto para p-excepcion como para l-excepcion. Esto nos indica que estos modelos se comportan como buenas medidas para determinar el crecimiento de las celdas afectadas, hasta el momento vimos revisiones con una regla.

4.3 Pruebas con una regla por revisión y aumento del N-Sparse

En esta sección afectamos a las pruebas con una variación en las características de los cubos, para ver que relación tiene con la cantidad de celdas que se afectan, la variable considerada es la que definimos anteriormente, llamada N-Sparse. Para estas pruebas utilizamos 70-Sparse, quiere decir que sólo el 30% de las combinaciones entre las dimensiones existen en el Fact table. Queremos estudiar si el que no existan algunas celdas con valores, afecta o no a la cantidad de celdas afectadas.

Dimension1-100-Level1 Dimension2-600- Level5 70-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100, y definimos reglas que vayan tomando distintos valores de *Paths revisados* entre el 0% y el 10% aproximadamente. El TargetLevel de la Dimension2 será como en la primera sección, Level5, para obtener un bajo nivel de celdas afectadas. Se varía el valor de Sparse que en este caso será de 70.

Dimension1

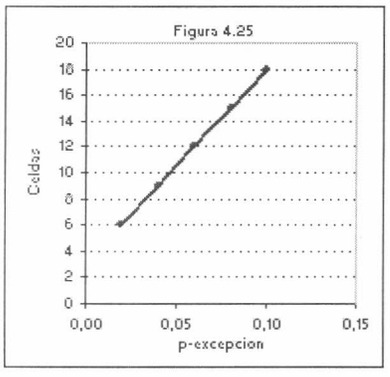
All	Level3	Level2	**Level1**	Level0
	13	26	51	100

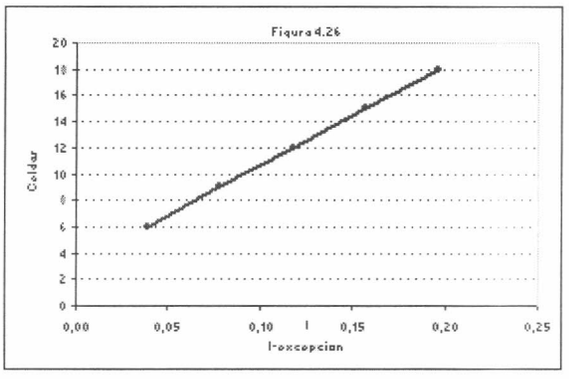
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	199	600

TabName's: Cubo0021_1 - Cubo0021_2 - Cubo0021_3 - Cubo0021_4 - Cubo0021_5
Cubo: Sales100_600_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0021_1	2	100	2	6	0,020	0,020	0,039	0,00	872,00
Cubo0021_2	4	100	3	9	0,040	0,040	0,078	0,00	611,00
Cubo0021_3	6	100	4	12	0,060	0,060	0,118	0,00	441,00
Cubo0021_4	8	100	5	15	0,080	0,080	0,157	0,00	651,00
Cubo0021_5	10	100	6	18	0,100	0,100	0,196	0,00	721,00





Dimension1-300-Level1 Dimension2-700- Level5 70-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 300.

Dimension1

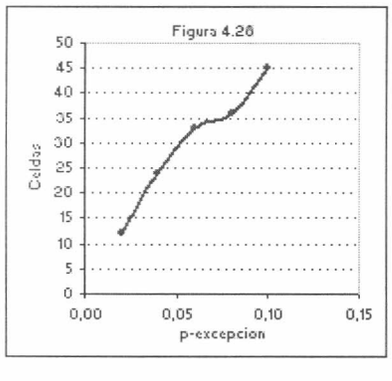
All	Level3	Level2	**Level1**	Level0
	38	76	151	300

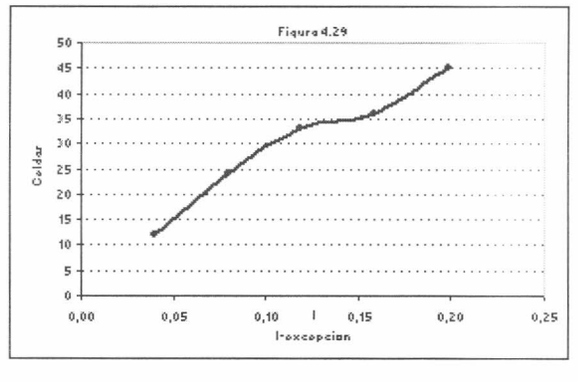
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	231	700

TabName's: Cubo0022_1 - Cubo0022_2 - Cubo0022_3 - Cubo0022_4 - Cubo0022_5
Cubo: Sales300_700_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0022_1	6	300	4	12	0,020	0,020	0,040	0,00	741,00
Cubo0022_2	12	300	8	24	0,040	0,040	0,079	0,00	871,00
Cubo0022_3	18	300	11	33	0,060	0,060	0,119	0,00	661,00
Cubo0022_4	24	300	12	36	0,080	0,080	0,159	0,00	811,00
Cubo0022_5	30	300	15	45	0,100	0,100	0,199	0,00	691,00





Dimension1-500-Level1 Dimension2-900- Level5 70-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 500.

Dimension1

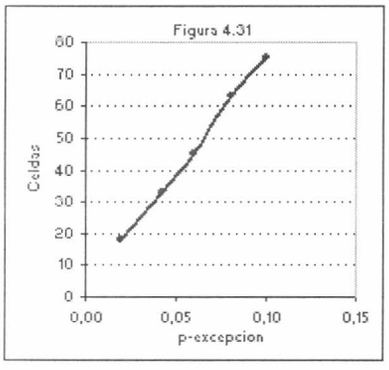
All	Level3	Level2	**Level1**	Level0
	63	126	251	500

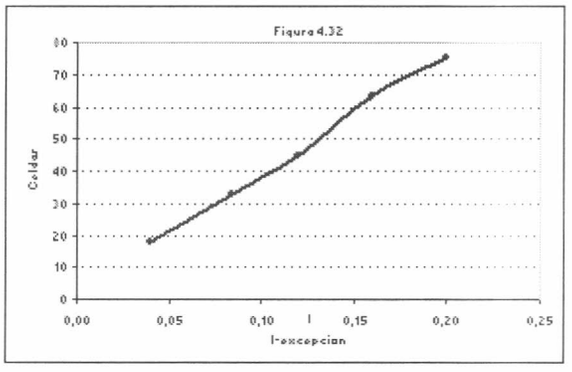
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	295	900

TabName's: Cubo0023_1 - Cubo0023_2 - Cubo0023_3 - Cubo0023_4 - Cubo0023_5
Cubo: Sales500_900_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0023_1	10	500	6	18	0,020	0,020	0,040	0,00	1091,00
Cubo0023_2	21	500	11	33	0,042	0,042	0,084	0,00	1132,00
Cubo0023_3	30	500	15	45	0,060	0,060	0,120	0,00	901,00
Cubo0023_4	40	500	21	63	0,080	0,080	0,159	0,00	1212,00
Cubo0023_5	50	500	25	75	0,100	0,100	0,199	0,00	1012,00





Dimension1-2000-Level1 Dimension2-900- Level5 70-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 2000.

Dimension1

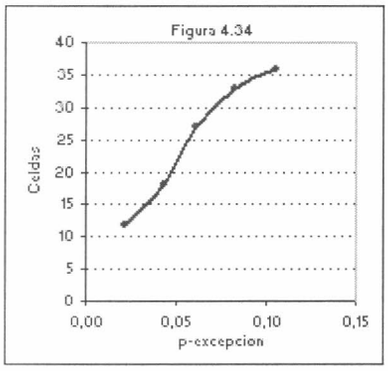
All	Level3	Level2	**Level1**	Level0
	251	501	1001	2000

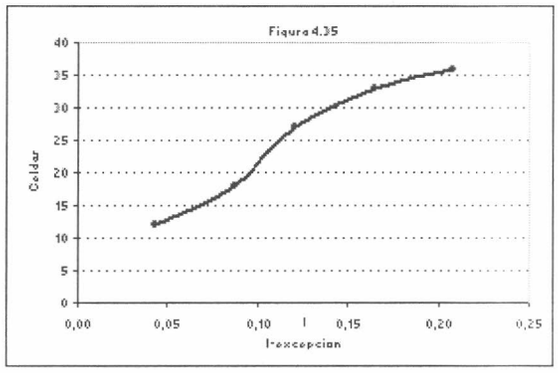
Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	295	900

TabName's: Cubo0024_1 - Cubo0024_2 - Cubo0024_3 - Cubo0024_4 - Cubo0024_5
Cubo: Sales2000_900_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0024_1	44	2000	4	12	0,022	0,022	0,04	0,00	1221,00
Cubo0024_2	88	2000	6	18	0,044	0,044	0,09	0,00	1603,00
Cubo0024_3	122	2000	9	27	0,061	0,061	0,12	0,00	1192,00
Cubo0024_4	166	2000	11	33	0,083	0,083	0,16	0,00	1172,00
Cubo0024_5	210	2000	12	36	0,105	0,105	0,21	0,00	1111,00





En los gráficos de esta sección vemos que las curvas no varían significativamente, ya que en muchos casos las celdas que en el cubo original no tenían contenido pueden obtener nuevos valores a partir de la revisión.

4.4 Pruebas con una regla por revisión y aumento del N-Sparse con más celdas afectadas.

Presentamos una prueba similar a la anterior pero aumentando el volumen de las instancias de las dimensiones, para estudiar como afecta esto a la característica de Sparse.

Dimension1-100-Level1 Dimension2-600- Level1 70-Sparse

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100. El TargetLevel de la Dimension2 será Level1 para obtener mayor número de celdas afectadas. También consideramos el valor de Sparse que será de 70.

Dimension1

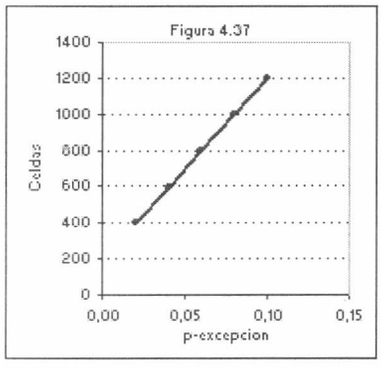
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
13 26 51 100

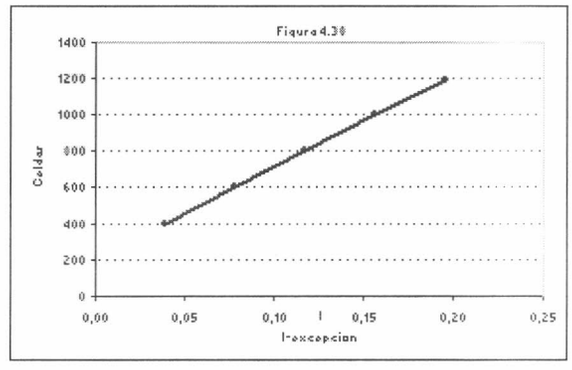
Dimension2

All — Level5 — Level4 — Level3 — Level2 — **Level1** — Level0
3 23 55 102 199 600

TabName's: Cubo0031_1 - Cubo0031_2 - Cubo0031_3 - Cubo0031_4 - Cubo0031_5
Cubo: Sales100_600_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0031_1	2	100	2	398	0,020	0,020	0,039	0,00	601,00
Cubo0031_2	4	100	3	597	0,040	0,040	0,078	0,00	620,00
Cubo0031_3	6	100	4	796	0,060	0,060	0,118	0,00	651,00
Cubo0031_4	8	100	5	995	0,080	0,080	0,157	0,00	641,00
Cubo0031_5	10	100	6	1194	0,100	0,100	0,196	0,00	651,00





Dimension1-300-Level1 Dimension2-700- Level1 70-Sparse

Para la dimensión revisada, tomamos un valor de *Paths Totales* de 300.

Dimension1

All — Level3 — Level2 — **Level1** — Level0
38 76 151 300

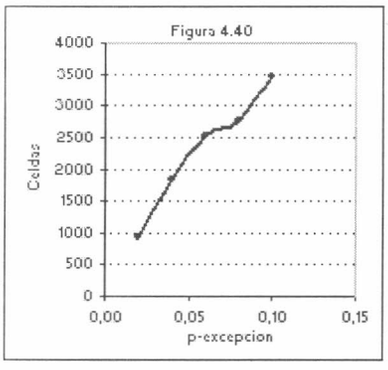
Dimension2

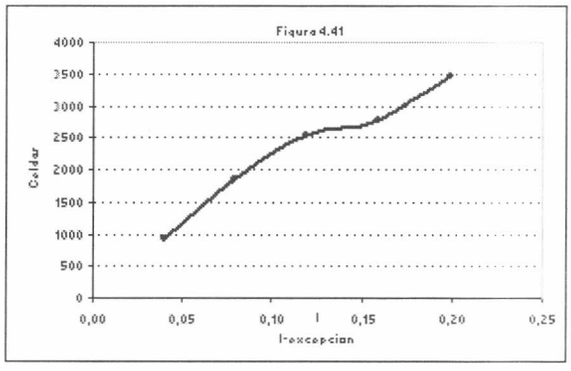
All — Level5 — Level4 — Level3 — Level2 — **Level1** — Level0
3 23 55 102 231 700

TabName's: Cubo0032_1 - Cubo0032_2 - Cubo0032_3 - Cubo0032_4 - Cubo0032_5
Cubo: Sales300_700_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0032_1	6	300	4	924	0,020	0,020	0,040	0,00	671,00
Cubo0032_2	12	300	8	1848	0,040	0,040	0,079	0,00	651,00

Cubo0032_3	18	300	11	2541	0,060	0,060	0,119	0,00	711,00
Cubo0032_4	24	300	12	2772	0,080	0,080	0,159	0,00	731,00
Cubo0032_5	30	300	15	3465	0,100	0,100	0,199	0,00	681,00





Dimension1-500-Level1 Dimension2-900- Level1 70-Sparse

Para la dimensión revisada, tomamos un valor de *Paths Totales* de 500.

Dimension1

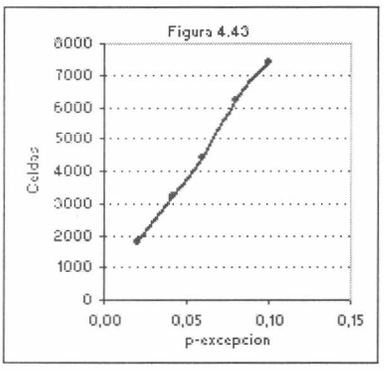
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
63 126 251 500

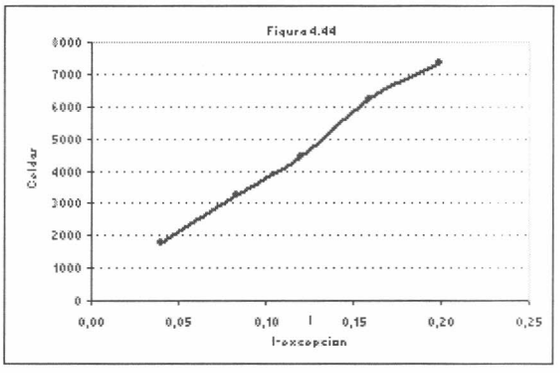
Dimension2

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 295 900

TabName's: Cubo0033_1 - Cubo0033_2 - Cubo0033_3 - Cubo0033_4 - Cubo0033_5
Cubo: Sales500_900_70
Revisiones: Rev0010 - Rev0014

Tabéame	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0033_1	10	500	6	1770	0,020	0,020	0,040	0,00	1662,00
Cubo0033_2	21	500	11	3245	0,042	0,042	0,084	0,00	1022,00
Cubo0033_3	30	500	15	4425	0,060	0,060	0,120	0,00	1583,00
Cubo0033_4	40	500	21	6195	0,080	0,080	0,159	0,00	1392,00
Cubo0033_5	50	500	25	7375	0,100	0,100	0,199	0,00	1242,00





Dimension1-2000-Level1 Dimension2-900- Level1 70-Sparse

Por último, para la dimensión revisada, tomamos un valor de *Paths Totales* de 2000.

Dimension1

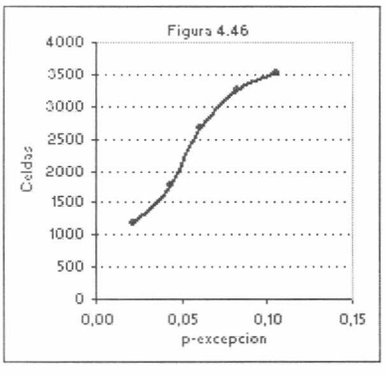
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
251 501 1001 2000

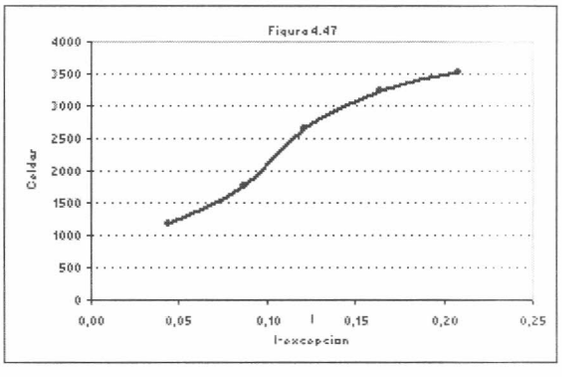
Dimension2

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 295 900

TabName's: Cubo0034_1 - Cubo0034_2 - Cubo0034_3 - Cubo0034_4 - Cubo0034_5
Cubo: Sales2000_900_70
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0034_1	44	2000	4	1180	0,022	0,022	0,04	0,00	912,00
Cubo0034_2	88	2000	6	1770	0,044	0,044	0,09	0,00	1191,00
Cubo0034_3	122	2000	9	2655	0,061	0,061	0,12	0,00	1352,00
Cubo0034_4	166	2000	11	3245	0,083	0,083	0,16	0,00	1122,00
Cubo0034_5	210	2000	12	3540	0,105	0,105	0,21	0,00	1182,00





En los cuatro gráficos anteriores vemos que para estos casos, las curvas no varían significativamente, siempre mantienen un ritmo creciente, como en las primeras cuatro pruebas que se hicieron considerando un 70-Sparse. Para estos casos, los resultados se deben a que las celdas afectadas se obtienen de los datos de los miembros de las dimensiones, no del contenido del cubo, igualmente suponiamos que podían variar significativamente las celdas afectadas, pero pasa que aunque algunas Celdas no tengan contenido, igualmente son afectadas por la revisión. Debido a este resultado en adelante no consideraremos las diferencias de Sparse y dirigiremos la atención a otras variables, a menos que lo creamos útil para algún caso específico.

4.5 Pruebas con más de una regla por revisión con una dimensión revisada.

Vimos que con una regla por revisión las curvas mantuvieron un comportamiento estable, ya que los modelos indicaban un crecimiento lineal de las celdas afectadas a medida que aumentaba la selectividad.

Ahora, para ahondar en detalles, consideramos revisiones con más de una regla definida por revisión. Esto nos da mayor rango de valores sobre todo en las pruebas con l-excepcion. En las siguientes pruebas agregamos gráficos que consideran el modelo f-excepcion.

Análisis a realizar

A continuación haremos testeos sobre los mismos cubos que en las pruebas anteriores, pero variando las revisiones. Estas contendrán más de una regla, por lo que se verán afectados varios niveles por las cláusulas WHERE y existirán varias cláusulas SET.

Debido a la naturaleza del cambio utilizamos el modelo l-excepcion y el nuevo modelo f-excepcion, ya que considera variables inherentes a los niveles en las dimensiones y cardinalidad de los niveles, no sólo los paths involucrados. De todos modos continuamos mostrando el comportamiento del modelo p-excepcion.

Los valores de cota para f-excepcion se fijarán según se vayan desarrollando las curvas, cuando estas vayan marcando una tendencia. Las cotas se basarán en la de las l-excepciones, que tomarán valores entre 0 y 0,20, estos valores dieron resultados satisfactorios en las pruebas anteriores. La cota de p-excepcion se fijará siguiendo el progreso de la cota de l-excepcion.

En todos los casos utilizamos las revisiones Rev0030-Rev0034 definidas al comienzo de este capítulo.

Dimension1-100-Level3 Dimension2-600- Level5 0-Sparse 3 Reglas

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100. El TargetLevel para la Dimension1 es el superior (Level3), ya que hay reglas definidas para todos los demás niveles inferiores. El TargetLevel de la Dimension2, es uno superior con baja cardinalidad en el nivel, para obtener valores bajos de celdas afectadas. Las reglas tienen cláusulas WHERE sobre los niveles 0, 1 y 2 y cláusulas SET sobre los niveles 1, 2 y 3 para hacer más variables las condiciones de los modelos l-excepcion y f-excepcion.

Dimension1

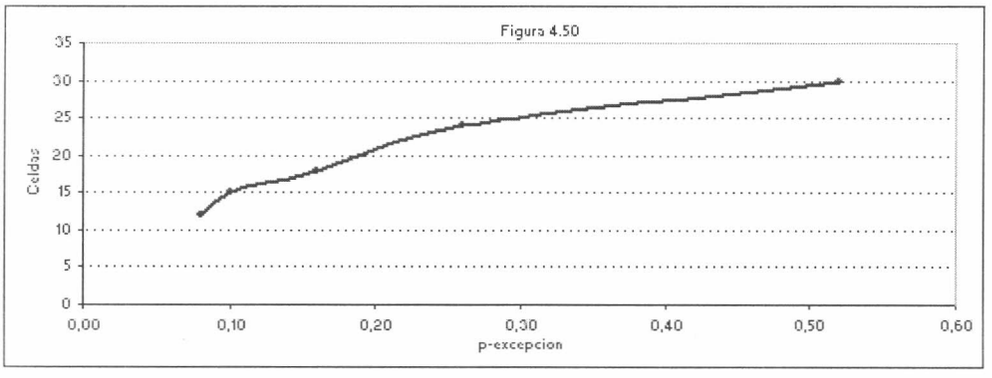
All	**Level3**	Level2	Level1	Level0
	13	26	51	100

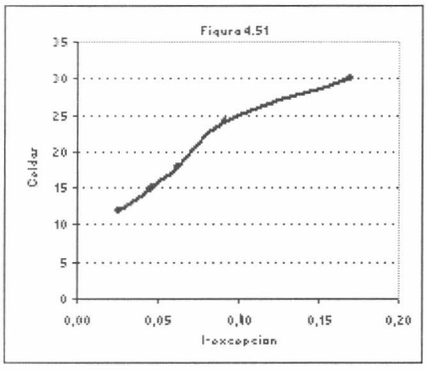
Dimension2

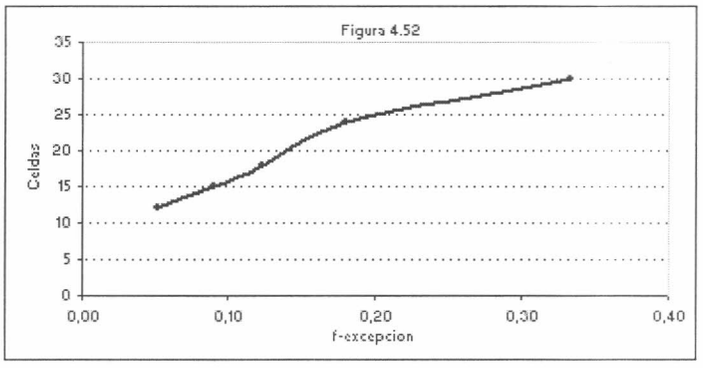
All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	199	600

TabName's: Cubo0041_1 - Cubo0041_2 - Cubo0041_3 - Cubo0041_4 - Cubo0041_5
Cubo: Sales100_600_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0041_2	8	100	4	12	0,080	0,026	0,052	0,00	651,00
Cubo0041_4	10	100	5	15	0,100	0,046	0,091	0,00	721,00
Cubo0041_5	16	100	6	18	0,160	0,063	0,123	0,00	711,00
Cubo0041_3	26	100	8	24	0,260	0,092	0,180	0,00	761,00
Cubo0041_1	52	100	10	30	0,520	0,170	0,334	0,00	991,00







Dimension1-300-Level3 Dimension2-700- Level5 0-Sparse 3 Reglas

Tomamos un valor de *Paths Totales* de 300 y vemos como se comporta con niveles de mayor cardinalidad, no sólo aumentamos la cantidad de los niveles inferiores sino de todos los niveles.

Dimension1

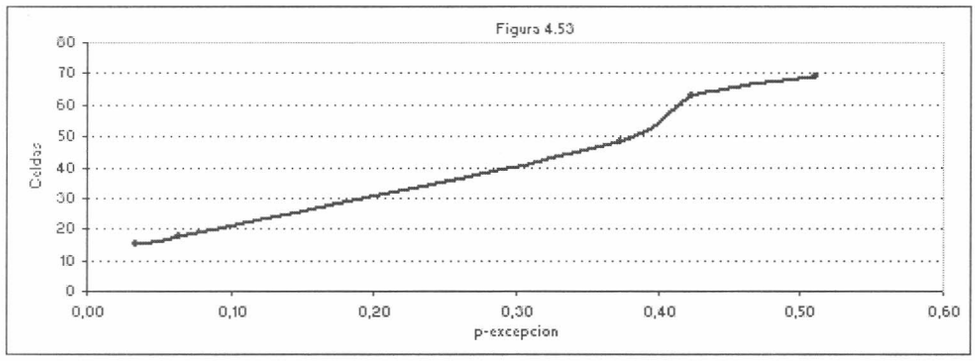
All — **Level3** — Level2 — Level1 — Level0
38 76 151 300

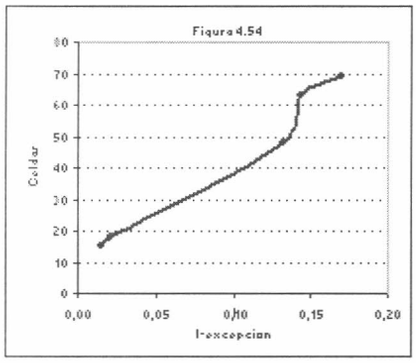
Dimension2

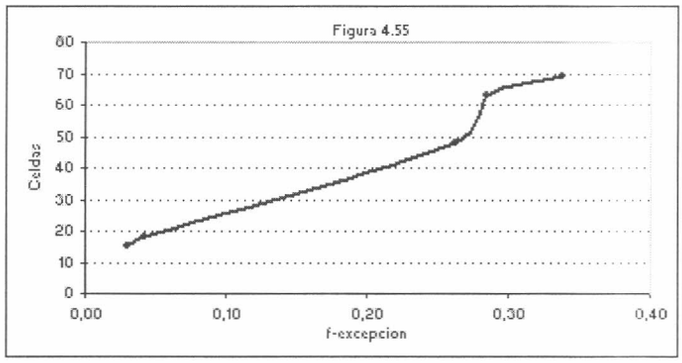
All — **Level5** — Level4 — Level3 — Level2 — Level1 — Level0
3 23 55 102 231 700

TabName's: Cubo0042_1 - Cubo0042_2 - Cubo0042_3 - Cubo0042_4 - Cubo0042_5
Cubo: Sales300_700_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0042_4	10	300	5	15	0,033	0,015	0,031	0,00	731,00
Cubo0042_5	19	300	6	18	0,063	0,021	0,042	0,00	811,00
Cubo0042_2	112	300	16	48	0,373	0,132	0,263	0,00	911,00
Cubo0042_3	127	300	21	63	0,423	0,144	0,285	0,00	1121,00
Cubo0042_1	153	300	23	69	0,510	0,170	0,338	0,00	1101,00







Dimension1-500-Level3 Dimension2-900- Level5 0-Sparse 3 Reglas

Variamos la cantidad de miembros de cada nivel, para ver como se comportan las curvas.

Dimension1

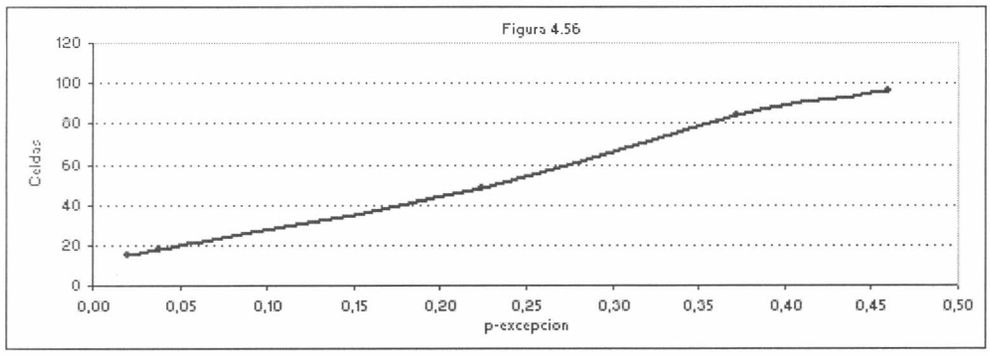
All	**Level3**	Level2	Level1	Level0
	63	126	251	500

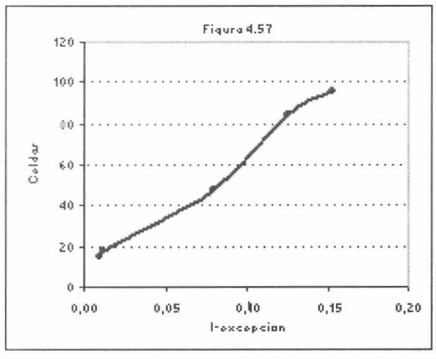
Dimension2

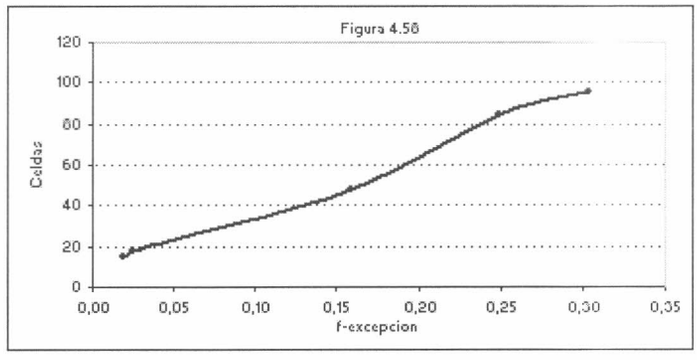
All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	295	900

TabName's: Cubo0043_1 - Cubo0043_2 - Cubo0043_3 - Cubo0043_4 - Cubo0043_5
Cubo: Sales500_900_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0043_4	10	500	5	15	0,020	0,009	0,019	0,00	831,00
Cubo0043_5	19	500	6	18	0,038	0,013	0,025	0,00	1251,00
Cubo0043_2	112	500	16	48	0,224	0,080	0,159	0,00	1292,00
Cubo0043_3	186	500	28	84	0,372	0,125	0,249	0,00	1161,00
Cubo0043_1	230	500	32	96	0,460	0,153	0,304	0,00	1222,00







Dimension1-2000-Level3 Dimension2-900- Level5 0-Sparse 3 Reglas

Aumentamos a 2000 la cantidad del nivel inferior de la dimensión revisada.

Dimension1

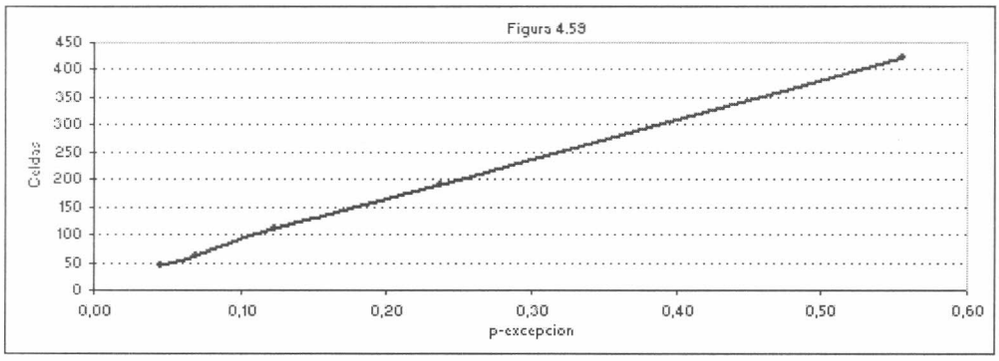
All ——— **Level3** ——— Level2 ——— Level1 ——— Level0
251 501 1001 2000

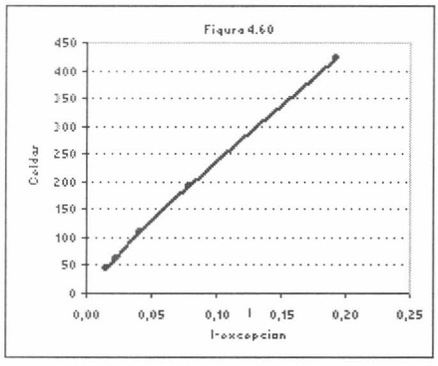
Dimension2

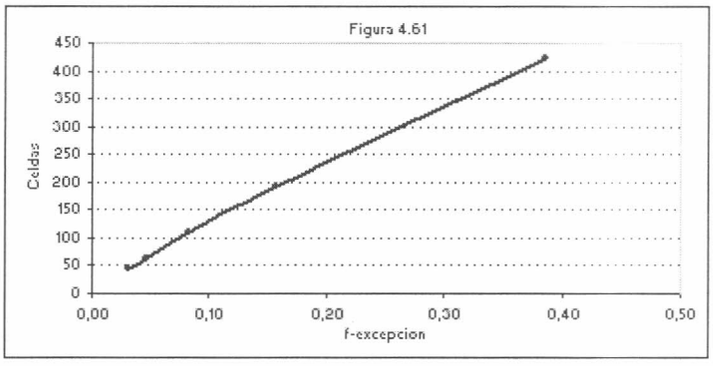
All ——— **Level5** ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0
3 23 55 102 295 900

TabName's: Cubo0044_1 - Cubo0044_2 - Cubo0044_3 - Cubo0044_4 - Cubo0044_5
Cubo: Sales2000_900_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0044_4	90	2000	15	45	0,045	0,016	0,031	0,00	1903,00
Cubo0044_5	139	2000	21	63	0,070	0,023	0,046	0,00	1672,00
Cubo0044_3	247	2000	37	111	0,124	0,042	0,083	0,00	2334,00
Cubo0044_1	473	2000	64	192	0,237	0,079	0,158	0,00	3675,00
Cubo0044_2	1112	2000	141	423	0,556	0,193	0,386	0,00	5087,00







Los resultados de los gráficos de las p-excepciones confirman su comportamiento, creciendo de forma lineal en cantidad de celdas a medida que crece la cota de p-excepcion.

Los resultados de l-excepcion muestran un comportamiento similar, aunque este modelo es más preciso para este caso ya que hay reglas en varios niveles, y se consideran variables independientes para cada nivel.

Las curvas de f-excepcion muestran un crecimiento menos pronunciado, pero siempre creciente, esto se debe a la dispersión que produce el factor FIO, en los valores de la f-excepcion.

En las próximas pruebas veremos, que sucede cuando aumenta la cantidad de celdas afectadas. Aunque ya variamos las principales variables de las entidades involucradas en la revisión, que son las revisiones con sus reglas y las instancias de las dimensiones y notamos que las curvas mantienen un comportamiento similar, lo que nos estaría indicando que los algoritmos son previsibles y podrían usarse sabiendo a priori como sería su comportamiento.

4.6 Pruebas con más de una regla por revisión con una dimensión revisada y mayor número celdas afectadas

En esta sección, aumentamos el número de celdas para ver como afecta a las curvas, especialmente a f-excepcion. Tomamos las mismas revisiones que en la sección anterior y consideramos las revisiones Rev0030 a Rev0034. Variamos el TargetLevel de la Dimension2 a un nivel inferior para obtener mayor cantidad de celdas afectadas..

Dimension1-100-Level3 Dimension2-600- Level1 0-Sparse 3 Reglas

Para la dimensión revisada tomamos un valor de *Paths Totales* de 100. El TargetLevel para la Dimension1 (revisada) será Level3, mientras que bajamos el TargetLevel de la Dimension2 a Level1 para obtener más Celdas afectadas.

Dimension1

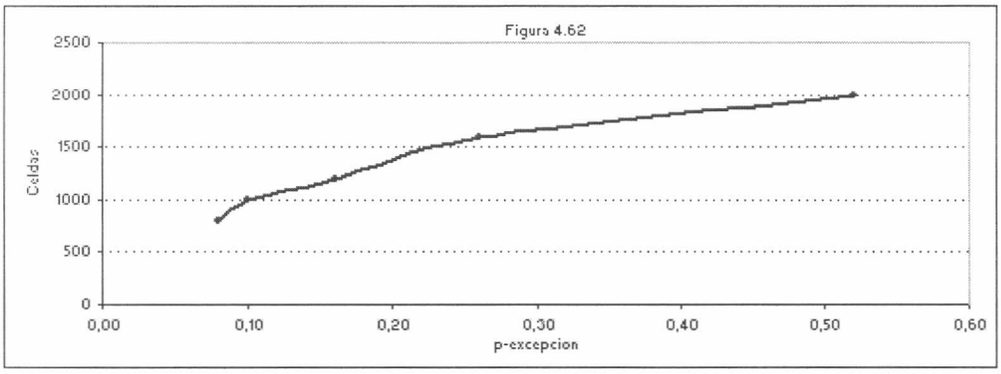
All	**Level3**	Level2	Level1	Level0
	13	26	51	100

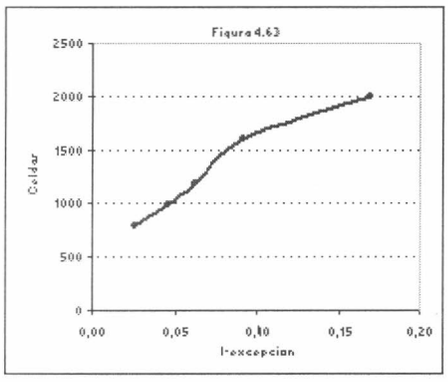
Dimension2

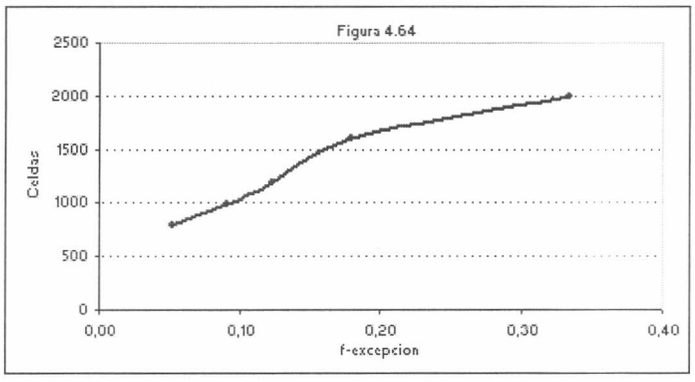
All	Level5	Level4	Level3	Level2	**Level1**	Level0
	3	23	55	102	199	600

TabName's: Cubo0051_1 - Cubo0051_2 - Cubo0051_3 - Cubo0051_4 - Cubo0051_5
Cubo: Sales100_600_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0051_2	8	100	4	796	0,080	0,026	0,052	0,00	661,00
Cubo0051_4	10	100	5	995	0,100	0,046	0,091	0,00	631,00
Cubo0051_5	16	100	6	1194	0,160	0,063	0,123	0,00	641,00
Cubo0051_3	26	100	8	1592	0,260	0,092	0,180	0,00	671,00
Cubo0051_1	52	100	10	1990	0,520	0,170	0,334	0,00	771,00







Dimension1-300-Level3 Dimension2-700- Level1 0-Sparse 3 Reglas

Aumentamos la cantidad de miembros de los niveles para ver como se comportan las curvas. En este caso tenemos 300 miembros para el nivel inferior.

Dimension1

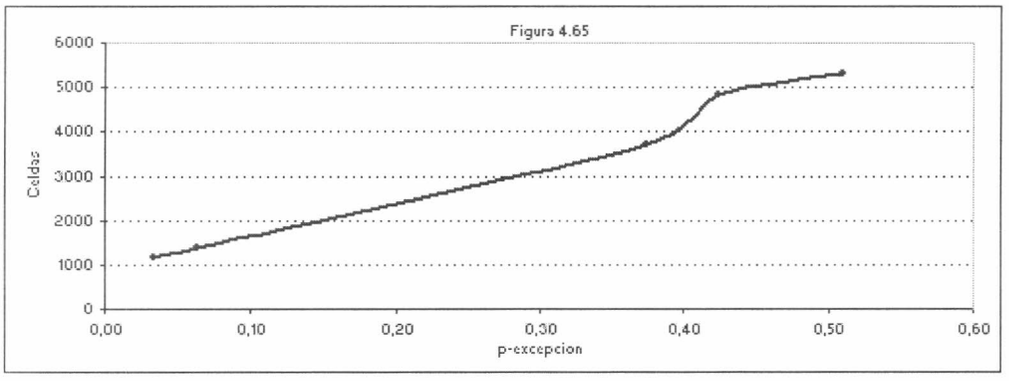
All ——— **Level3** ——— Level2 ——— Level1 ——— Level0
38 76 151 300

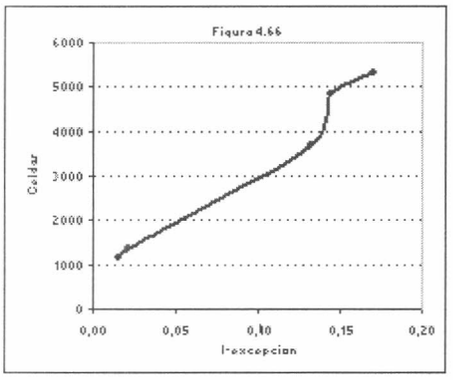
Dimension2

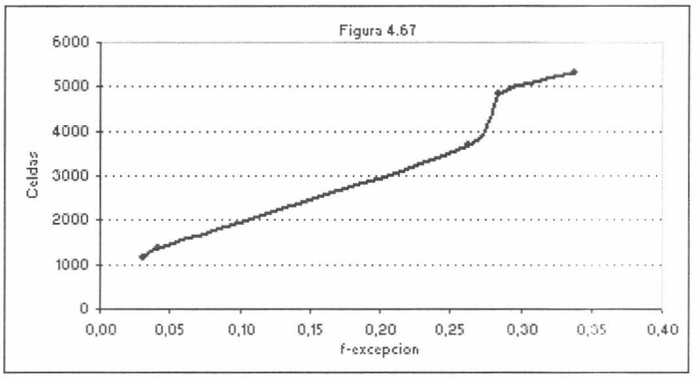
All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 231 700

TabName's: Cubo0052_1 - Cubo0052_2 - Cubo0052_3 - Cubo0052_4 - Cubo0052_5
Cubo: Sales300_700_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0052_4	10	300	5	1155	0,033	0,015	0,031	0,00	902,00
Cubo0052_5	19	300	6	1386	0,063	0,021	0,042	0,00	902,00
Cubo0052_2	112	300	16	3696	0,373	0,132	0,263	0,00	821,00
Cubo0052_3	127	300	21	4851	0,423	0,144	0,285	0,00	891,00
Cubo0052_1	153	300	23	5313	0,510	0,170	0,338	0,00	981,00







Dimension1-500-Level3 Dimension2-900- Level1 0-Sparse 3 Reglas

Seguimos aumentando la cantidad de miembros de los niveles para ver como se comportan las curvas. En este caso 500 miembros para el nivel inferior.

Dimension1

All — **Level3** — Level2 — Level1 — Level0
63 126 251 500

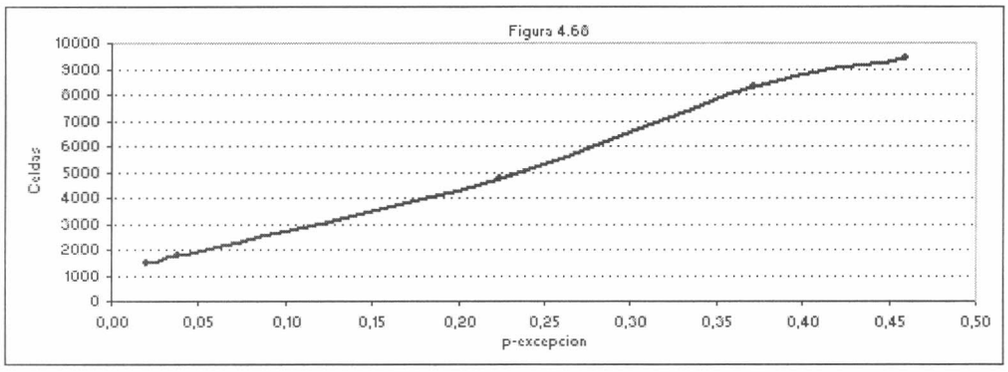
Dimension2

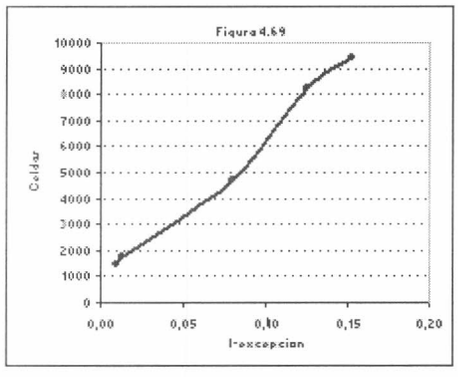
All — Level5 — Level4 — Level3 — Level2 — **Level1** — Level0
3 23 55 102 295 900

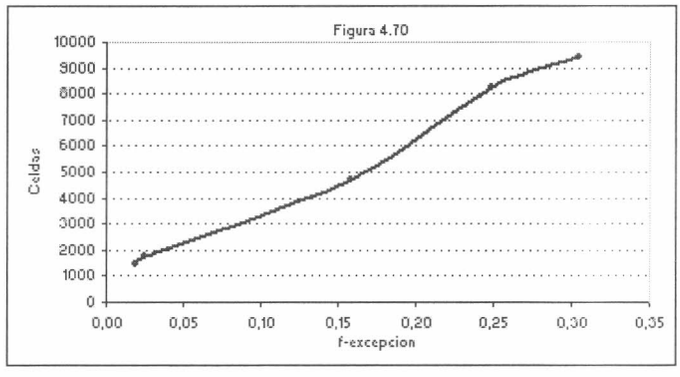
TabName's: Cubo0053_1 - Cubo0053_2 - Cubo0053_3 - Cubo0053_4 - Cubo0053_5
Cubo: Sales500_900_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0053_4	10	500	5	1475	0,020	0,009	0,019	0,00	1152,00
Cubo0053_5	19	500	6	1770	0,038	0,013	0,025	0,00	972,00
Cubo0053_2	112	500	16	4720	0,224	0,080	0,159	0,00	1002,00
Cubo0053_3	186	500	28	8260	0,372	0,125	0,249	0,00	1142,00

Cubo0053_1	230	500	32	9440	0,460	0,153	0,304	0,00	1362,00







Dimension1-2000-Level3 Dimension2-900- Level1 0-Sparse 3 Reglas

En este caso contamos con 2000 miembros para el nivel inferior, mientras que la distribución de miembros en los niveles las vemos en los gráficos de más abajo.

Dimension1

All ——— **Level3** ——— Level2 ——— Level1 ——— Level0
251 501 1001 2000

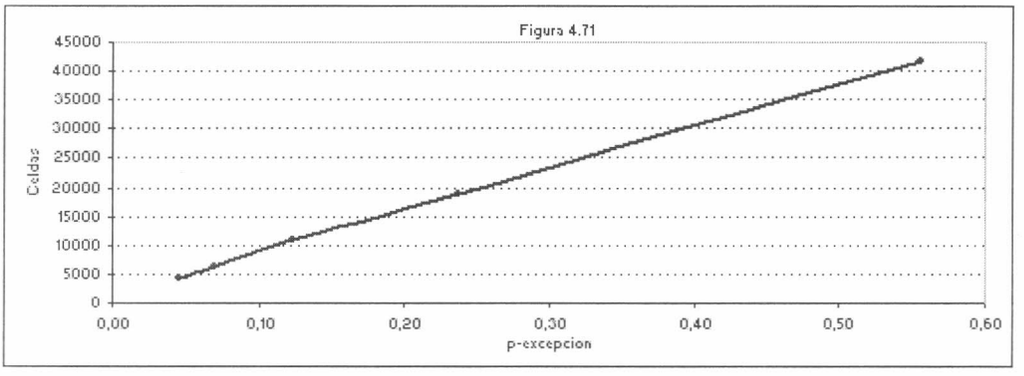
Dimension2

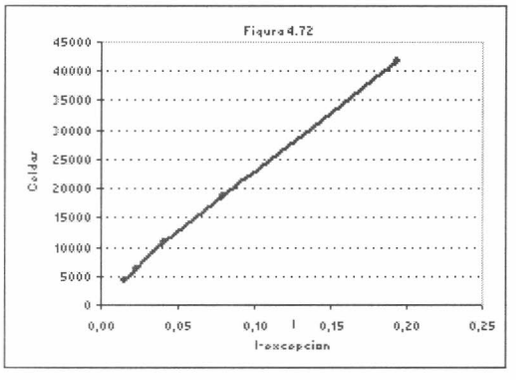
All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 29 900

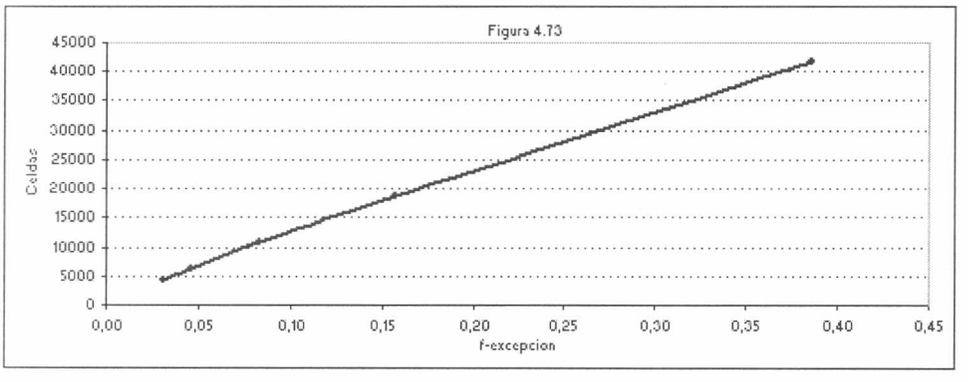
TabName's: Cubo0054_1 - Cubo0054_2 - Cubo0054_3 - Cubo0054_4 - Cubo0054_5
Cubo: Sales2000_900_0
Revisiones: Rev0030 - Rev0034

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	f-excep.	T-MDX (1)	T-Total
Cubo0054_4	90	2000	15	4425	0,045	0,016	0,031	0,00	1802,00

Cubo0054_5	139	2000	21	6195	0,070	0,023	0,046	0,00	1622,00
Cubo0054_3	247	2000	37	10915	0,124	0,042	0,083	0,00	2283,00
Cubo0054_1	473	2000	64	18880	0,237	0,079	0,158	0,00	3115,00
Cubo0054_2	1112	2000	141	41595	0,556	0,193	0,386	0,00	5407,00







Notamos que el aumento en la cantidad de celdas no afecta a las curvas, las mismas se comportan de manera similar bajo distintas condiciones de Sparse, cantidad de reglas y volumen. Concluimos que para casos de instancias similares a las testeadas y siguiendo los modelos de excepción, el comportamiento es aceptable, ya que las curvas crecen de manera lineal.

4.7 Pruebas con más de una dimensión revisada

En esta sección incluimos una variante más, es la cantidad de dimensiones que se revisan. Haremos las pruebas sobre cuatro de los cubos, revisando la Dimension1 con las revisiones Rev0010 a la Rev0014 y la Dimension2 con las revisiones RevProd30 a la RevProd34 definidas al comienzo del capítulo.

Las cinco revisiones que existen para cada dimensión, se asocian una a una de la siguiente manera:

Dimension1	Dimension2
Rev0010	**RevProd34**
Rev0011	**RevProd30**
Rev0012	**RevProd31**
Rev0013	**RevProd32**
Rev0014	**RevProd33**

Seteamos el TargetLevel de la Dimension1 al nivel Level1 y como TargetLevel de la Dimension2 al nivel Level3.

En estas pruebas consideramos los modelos p-excepcion y l-excepcion, obviamos las pruebas con f-excepcion ya que las revisiones consideradas contienen una regla cada una.

Debido a que las p-excepcion y l-excepcion están definidas para la revisión de una dimensión, creamos nuevos modelos basados en los originales para que consideren los valores cuando hay más de una dimensión revisada. A los modelos modificados las llamamos p'-excepcion y l'-excepcion que definimos a continuación:.

p'-excepciones

$$p'\text{-excepcion} = \frac{(\Sigma 1_{..n}(\text{cantidad de paths revisados en la dimensión n})}{(\Sigma 1_{..n}(\text{paths totales dimensión n})}$$

l'-excepciones

$$l'\text{-excepcion} = \frac{(\Sigma 1_{..n}(\Sigma_{0..L}(\text{Miembros selecc. del nivel L / Cant. miembros del nivel L})) \text{ p/c dimensión n})}{(\Sigma 1_{..n}(\text{Cant niveles revisados}) \text{ p/c dimensóon n})}$$

El proceso de revisión de distintas dimensiones es independiente para cada una, y cada revisión producirá su conjunto de celdas a revisar. De todos modos estudiamos como

se comportan las curvas de p'-excepcion y l'-excepcion en relación a las curvas originales de cada dimensión.

Mostramos dos gráficos por tabla, cada gráfico contendrá cuatro curvas, el primero muestra la p-excepcion de las dimensiones revisadas de forma independiente, la p'-excepcion, que es con las dos dimensiones revisadas al mismo tiempo y la suma real de las p-excepcion ejecutadas de forma independiente. El segundo gráfico muestra lo mismo que el primero, pero relacionados con l-excepcion y l'-excepcion.

Existen diferencias en los rangos de los modelos, ya que se tomaron pruebas anteriores para no modificar los parámetros.

Dimension1-100-Level1 1 Regla Dimension2-600- Level3 0-Sparse 2 Dimensiones

Dimension1

All	Level3	Level2	**Level1**	Level0
	13	26	51	100

Dimension2

All	Level5	Level4	**Level3**	Level2	Level1	Level0
	3	23	55	102	199	600

Tabéame	Dimension	PathsRevisados	PathsTotales	Coordenadas	AffCells	p-excep	l-excep	p'-excep	l'-excep
Cubo0301_1	Dimension1	1	100	2	212	0,010	0,010		
Cubo0301_1	Dimension2	2	600	2	212	0,003	0,010	0,0043	0,0099
Cubo0301_2	Dimension1	2	100	2	263	0,020	0,020		
Cubo0301_2	Dimension2	16	600	3	263	0,027	0,020	0,0257	0,0197
Cubo0301_3	Dimension1	3	100	3	369	0,030	0,030		
Cubo0301_3	Dimension2	18	600	4	369	0,030	0,029	0,0300	0,0296
Cubo0301_4	Dimension1	4	100	3	420	0,040	0,040		
Cubo0301_4	Dimension2	22	600	5	420	0,037	0,039	0,0371	0,0394
Cubo0301_5	Dimension1	5	100	4	526	0,050	0,050		
Cubo0301_5	Dimension2	28	600	6	526	0,047	0,049	0,0471	0,0493

A continuación mostramos una tabla con la información de las celdas afectadas y los valores de p-excepcion y l-excepcion para Dimension1 y Dimension2 ejecutándose de manera independiente. Y debajo se muestra una tabla con los promedios de los valores entre los modelos de las dos dimensiones, esto se hizo para tomar los puntos fijos de la curva de la Suma real, ya que los valores individuales de las dos dimensiones estaban cerca pero no eran el mismo, entonces la curva se alejaba un poco de la realidad, la curva que se muestra como *"Suma real"* es una aproximación a la suma real.

Dimension1			Dimension2		
AffCells	p-excep	l-excep	AffCells	p-excep	l-excep
110	0,0100	0,0100	102	0,0033	0,0098
110	0,0200	0,0200	153	0,0267	0,0196
165	0,0300	0,0300	204	0,0300	0,0294
165	0,0400	0,0400	255	0,0367	0,0392

220	0,0500	0,0500	306	0,0467	0,0490

AffCells	p-excep-avg = (p-excep-Dim1 + p-excep-Dim2) / 2	l-excep-avg = (l-excep-Dim1 + l-excep-Dim2) / 2
212	0,0067	0,0099
263	0,0233	0,0198
369	0,0300	0,0297
420	0,0383	0,0396
526	0,0483	0,0495

Gráfico de p-excepcion – p'-excepcion

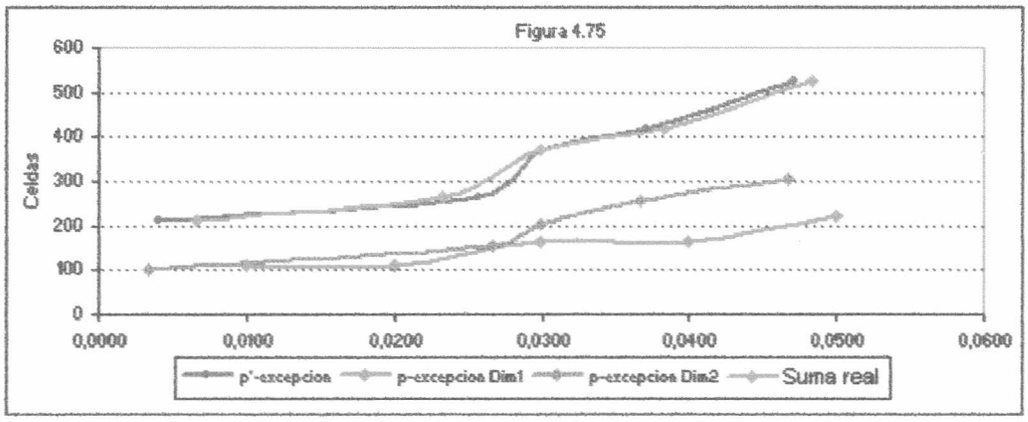
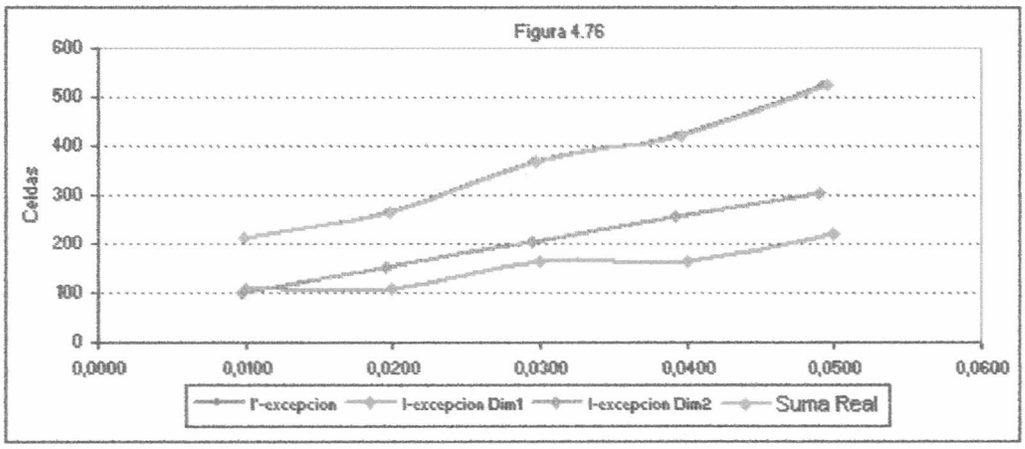


Gráfico de l-excepcion – l'-excepcion



Las curvas de los dos modelos se comportan de manera similar a las pruebas anteriores, con un crecimiento lineal. También se ve que la curva de los nuevos modelos, p'-excepcion y l'-excepcion abarcan un rango superior de celdas afectadas, ya que es la unión de las dos dimensiones, pero conserva la forma y crecimiento de las curvas de las revisiones independientes de las dimensiones. Las curvas de p'-excepcion y l'-excepcion están casi en la misma posición que la curva aproximada de la Suma Real.

Dimension1-2000-Level3 1 Regla Dimension2-900- Level3 0-Sparse 2 Dimensiones

Debido a que la prueba con el cubo más pequeño no tuvo variaciones significativas, vamos a testear con el más grande, salteando los dos cubos intermedios, para ver si hay un comportamiento similar. Acá usaremos las mismas revisiones que en el caso anterior, pero aumentando la cantidad de miembros de los niveles de ambas dimensiones.

Dimension1

All — Level3 — Level2 — **Level1** — Level0
251 501 1001 2000

Dimension2

All — Level5 — Level4 — **Level3** — Level2 — Level1 — Level0
3 23 55 102 295 900

TabName	PathsRevisados	PathsTotales	Coordenadas	AffCells	p-excep	l-excep	p'-excep	l'-excep
Cubo0304_1	22	2000	12	2662	0,01	0,01		
Cubo0304_1	2	900	2	2662	0,00	0,01	0,0066	0,0104
Cubo0304_2	33	2000	18	3993	0,02	0,02		
Cubo0304_2	20	900	3	3993	0,02	0,02	0,0194	0,0181
Cubo0304_3	44	2000	23	5269	0,02	0,02		
Cubo0304_3	23	900	4	5269	0,03	0,03	0,0238	0,0257
Cubo0304_4	55	2000	29	6600	0,03	0,03		
Cubo0304_4	29	900	5	6600	0,03	0,04	0,0299	0,0334
Cubo0304_5	66	2000	34	7876	0,03	0,03		
Cubo0304_5	38	900	6	7876	0,04	0,05	0,0376	0,0410

A continuación mostramos una tabla con la información de las celdas afectadas y los valores de p-excepcion y l-excepcion para Dimension1 y Dimension2, pero ejecutándose de manera independiente. Y debajo se muestra una tabla con los promedios de los valores entre los modelos de las dos dimensiones.

Dimension1			Dimension2		
AffCells	p-excep	l-excep	AffCells	p-excep	l-excep
660	0,0110	0,0110	2002	0,0022	0,0098
990	0,0165	0,0165	3003	0,0222	0,0196
1265	0,0220	0,0220	4004	0,0256	0,0294
1595	0,0275	0,0275	5005	0,0322	0,0392
1870	0,0330	0,0330	6006	0,0422	0,0490

AffCells	p-excep-avg = (p-excep-Dim1 + p-excep-Dim2) / 2	l-excep-avg = (l-excep-Dim1 + l-excep-Dim2) / 2
2662	0,0066	0,0104
3993	0,0194	0,0181
5269	0,0238	0,0257
6600	0,0299	0,0334
7876	0,0376	0,0410

Gráfico de l-excepcion – l'-excepcion

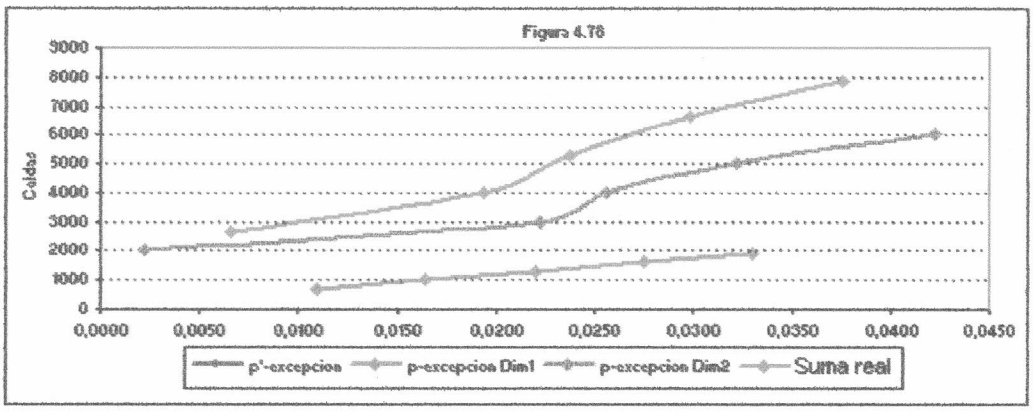
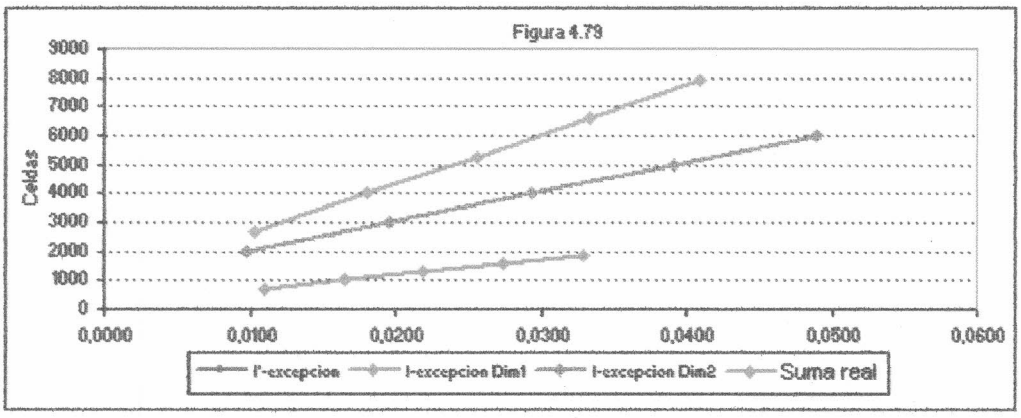


Gráfico de l-excepcion – l'-excepcion



Las revisiones de las distintas dimensiones involucradas son independientes, el comportamiento es similar al de sólo revisar una dimensión, la cardinalidad de la cantidad de celdas tampoco se ve afectada por la cantidad de dimensiones, ya que esto lo afecta el TargetLevel seleccionado para las dimensiones. La cantidad de celdas afectadas será aproximadamente la unión de las celdas afectadas que produzca la revisión de cada dimensión revisada de forma independiente. Para este último caso, las curvas de p'-excepcion y l'-excepcion casi no se ven porque las tapa la curva de la Suma real.

4.8 Análisis de tiempos

En este análisis, tomamos como variable en el eje de las Y el tiempo insumido en la ejecución de los MDX de recálculo y el tiempo total insumido en la revisión. Estos valores son fundamentales a la hora de determinar si una revisión es viable de ser realizada o no y también para determinar si puede ser ejecutada on-line o deberá ser ejecutada off-line. También servirá para estimar la potencia que deberá tener el hardware donde se ejecute la revisión y la escalabilidad de los algoritmos.

Para hacer este análisis, tomamos como muestra los primeros ocho cubos testeados en este capítulo en las secciones 4.1 y 4.2, y veremos como se comporta el

tiempo de procesamiento en las mismas cotas originales respecto a la cantidad de celdas afectadas.

Los cubos utilizados son desde el Cubo0001 al Cubo0004 los que tienen una cantidad baja de celdas afectadas, ya que el TargetLevel de Dimension1 es Level1 y el de Dimension2 es Level5 y desde el Cubo0011 al Cuvbo0014 que tienen una cantidad más alta de celdas afectadas, ya que el TargetLevel de Dimension1 es Level1 y el de Dimension2 es Level1.

Los gráficos muestran dos series en el eje de las Y, una es el *Tiempo MDX*, que indica el tiempo insumido en el recálculo de las celdas afectadas, este es el tiempo de la ejecución de los MDX puramente. La otra es el *Tiempo Total*, que es el tiempo total de procesamiento de la revisión. Veremos también que relación existe entre ambas curvas. En ambos casos .la unidad de medida en este eje será milisegundos.

Por una cuestión de claridad repetimos el contenido de las tablas con sus valores.

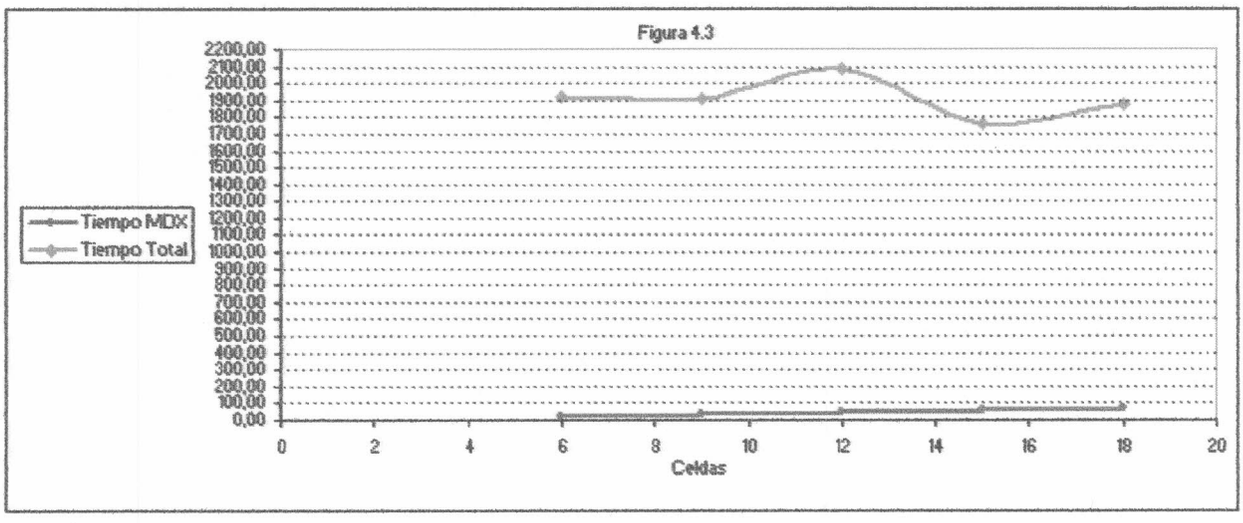
4.8.1 Pruebas con una regla por revisión y una dimensión revisada

Dimension1-100-Level1 Dimension2-600- Level5 0-Sparse

En estas cuatro pruebas incrementamos la cantidad de miembros de las dimensiones de la misma manera que se hizo en 4.1. Comenzamos por 100 miembros en el nivel inferior de Dimension1

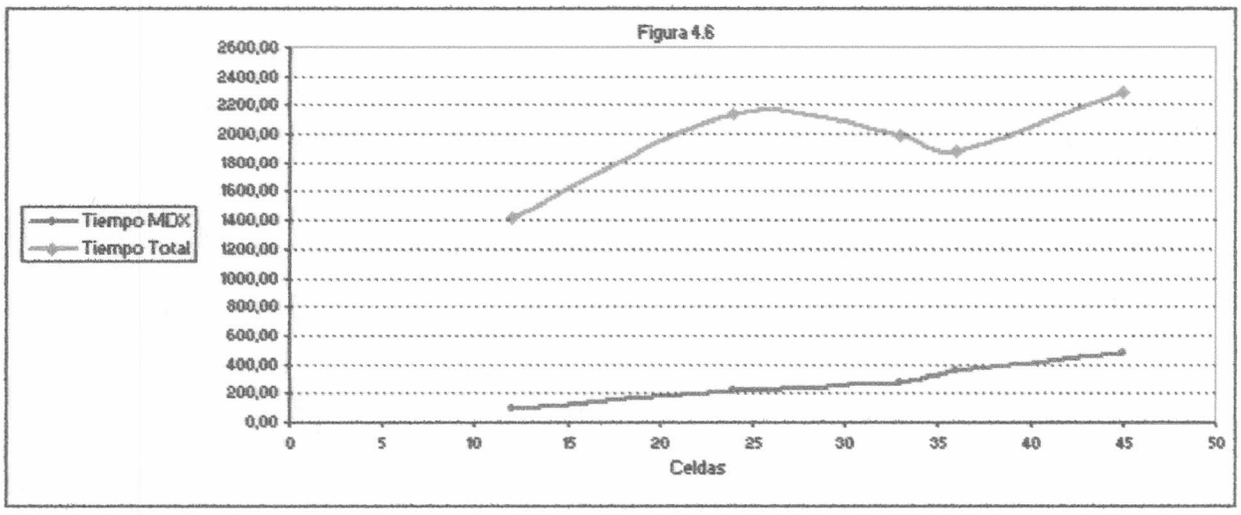
Tabéame	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0001_1	2	100	2	6	0,020	0,020	20,00	1923,00	92	300,00
Cubo0001_2	4	100	3	9	0,040	0,040	40,00	1913,00	184	225,00
Cubo0001_3	6	100	4	12	0,060	0,060	50,00	2083,00	276	240,00
Cubo0001_4	8	100	5	15	0,080	0,080	60,00	1763,00	368	250,00
Cubo0001_5	10	100	6	18	0,100	0,100	70,00	1872,00	460	257,14

Figura 4.3



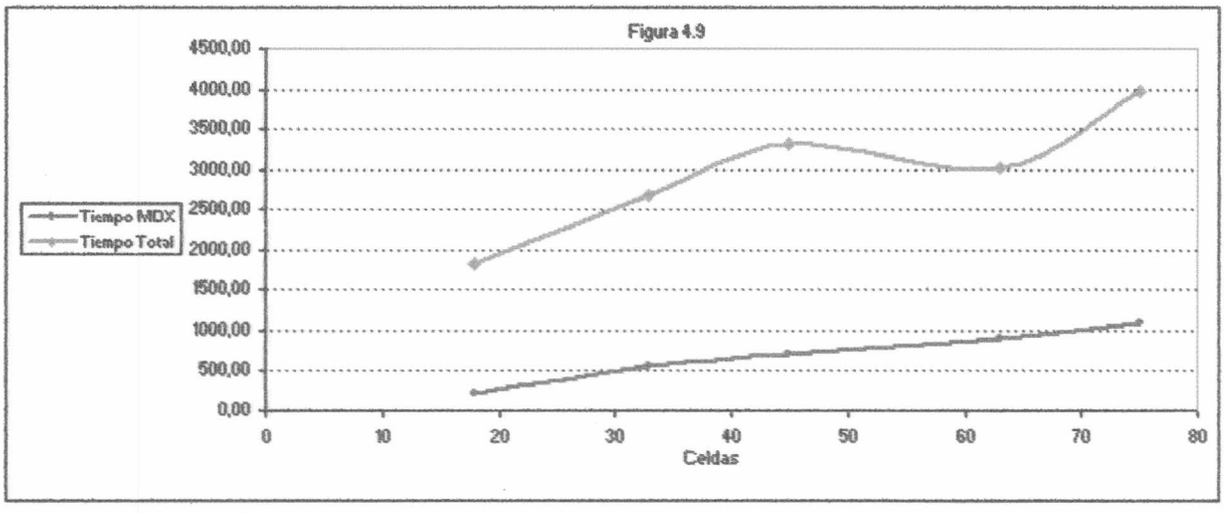
Dimension1-300-Level1 Dimension2-700- Level5 0-Sparse

Tabéame	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0002_1	6	300	4	12	0,020	0,020	100,00	1422,00	276	120,00
Cubo0002_2	12	300	8	24	0,040	0,040	220,00	2133,00	552	109,09
Cubo0002_3	18	300	11	33	0,060	0,060	270,00	1993,00	828	122,22
Cubo0002_4	24	300	12	36	0,080	0,080	360,00	1872,00	1012	100,00
Cubo0002_5	30	300	15	45	0,100	0,100	480,00	2293,00	1288	93,75



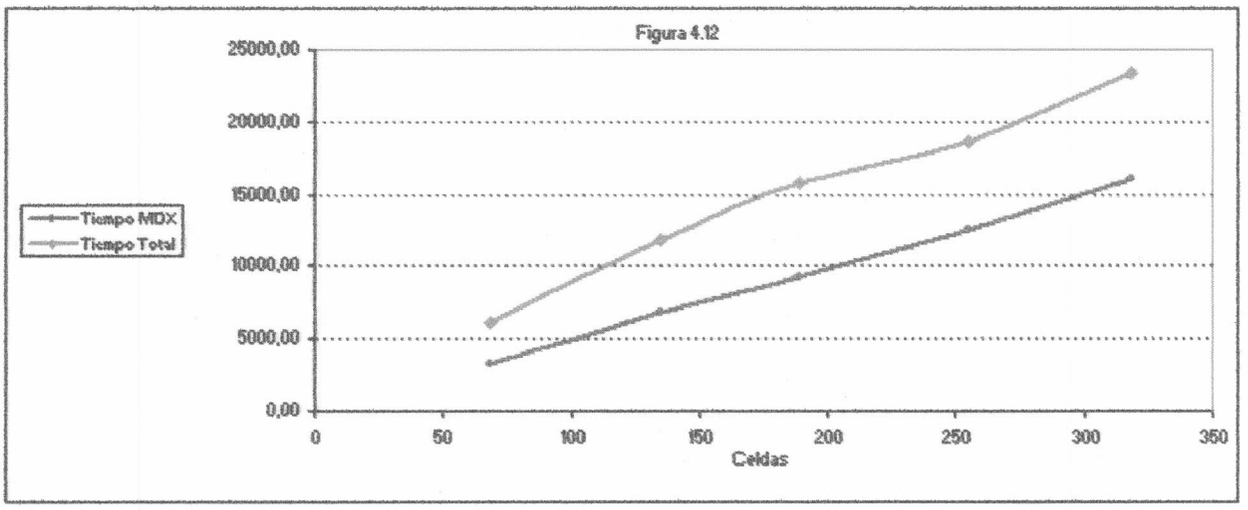
Dimension1-500-Level1 Dimension2-900- Level5 0-Sparse

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0003_1	10	500	6	18	0,020	0,020	211,00	1823,00	460	85,31
Cubo0003_2	21	500	11	33	0,042	0,042	561,00	2674,00	874	58,82
Cubo0003_3	30	500	15	45	0,060	0,060	711,00	3314,00	1288	63,29
Cubo0003_4	40	500	21	63	0,080	0,080	891,00	3014,00	1748	70,71
Cubo0003_5	50	500	25	75	0,100	0,100	1091,00	3955,00	2208	68,74



Dimension1-2000-Level1 Dimension2-900- Level5 0-Sparse

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0004_1	44	2000	23	69	0,022	0,022	3275,00	6129,00	2024	21,07
Cubo0004_2	88	2000	45	135	0,044	0,044	6810,00	11817,00	4048	19,82
Cubo0004_3	122	2000	63	189	0,061	0,061	9253,00	15792,00	5612	20,43
Cubo0004_4	166	2000	85	255	0,083	0,083	12559,00	18567,00	7636	20,30
Cubo0004_5	210	2000	106	318	0,105	0,105	15994,00	23314,00	9614	19,88



En estos cuatro gráficos notamos dos cosas importantes, una es que la variable pura del tiempo de recálculo de las celdas se comporta de manera similar a las curvas de los X-excepcion, proporcionalmente creciente.

Por otro lado vemos que a medida que el cubo va creciendo en volumen, y por ende crece la probabilidad de afectar más celdas, la curva del tiempo de procesamiento total se va acercando en forma y valor a la curva de tiempo insumido para el recálculo de las celdas. En las primeras pruebas notamos una mayor distancia entre las curvas de *Tiempo MDX* y *Tiempo total,* esto se debe a que hay pocas celdas que recalcular, entonces el tiempo de recálculo es pequeño comparado con el tiempo total del algoritmo, ya que hay tiempos fijos como los que el algoritmo consume en armar estructuras, grabar información estadística, almacenar datos para permitir la ejecución de las distintas partes del algoritmo, etc.

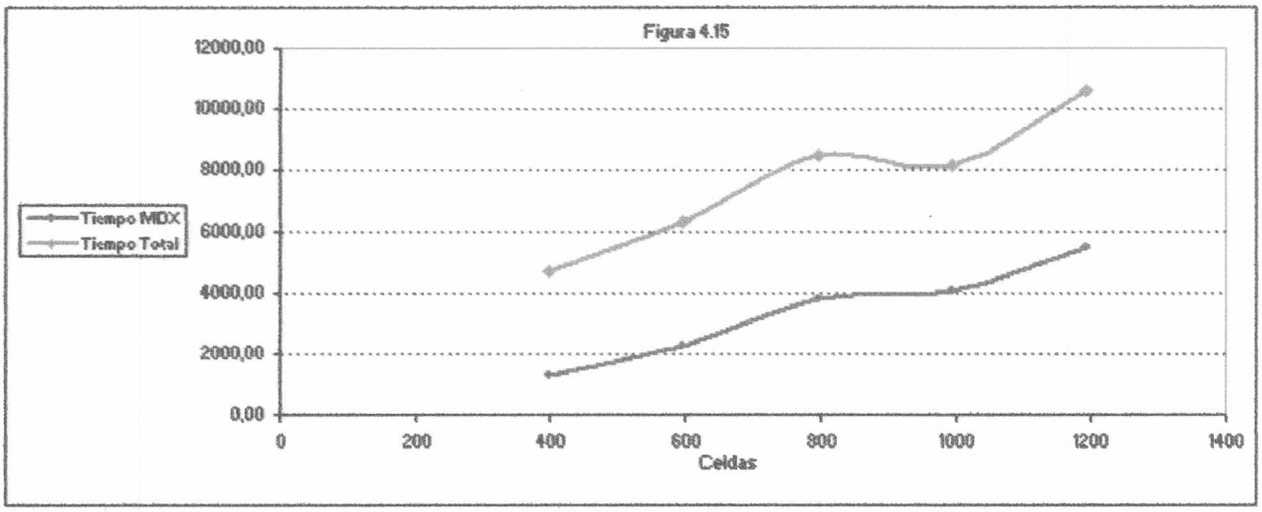
Los valores obtenidos muestran que la cantidad de Celdas afectadas es un buen valor para estimar el tiempo que llevará revisar un cubo.

4.8.2 Pruebas con una regla por revisión, una dimensión revisada y mayor cantidad de celdas afectadas

En esta sección haremos pruebas similares a las de la sección anterior, pero seteando el TargetLevel de la Dimension2 al Level1, para obtener más celdas afectadas y ver si las curvas siguen el mismo comportamiento. Queremos confirmar que la mayor cantidad de tiempo se insume en el recalculo puro de celdas.

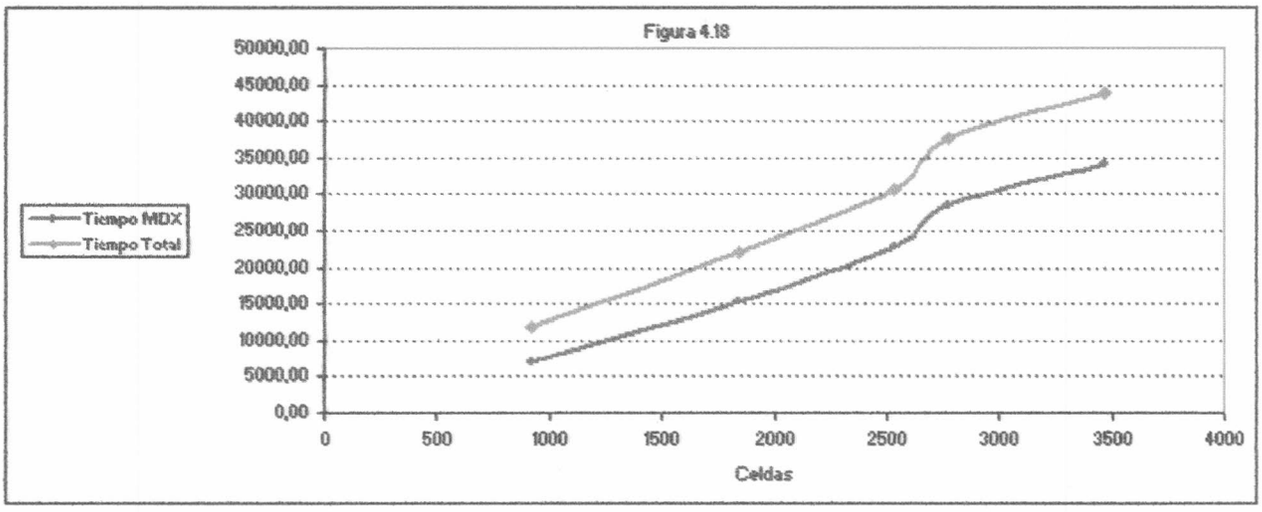
Dimension1-100-Level1 Dimension2-600- Level1 0-Sparse

Tabdame	PathsRevis.	Paths	Coords	AffCells	p-excep.	i-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0011_1	2	100	2	398	0,020	0,020	1302,00	4706,00	2400	305,68
Cubo0011_2	4	100	3	597	0,040	0,040	2244,00	6269,00	4800	266,60
Cubo0011_3	6	100	4	796	0,060	0,060	3766,00	8492,00	7200	211,36
Cubo0011_4	8	100	5	995	0,080	0,080	4047,00	8142,00	9600	245,90
Cubo0011_5	10	100	6	1194	0,100	0,100	5470,00	10596,00	12000	218,28



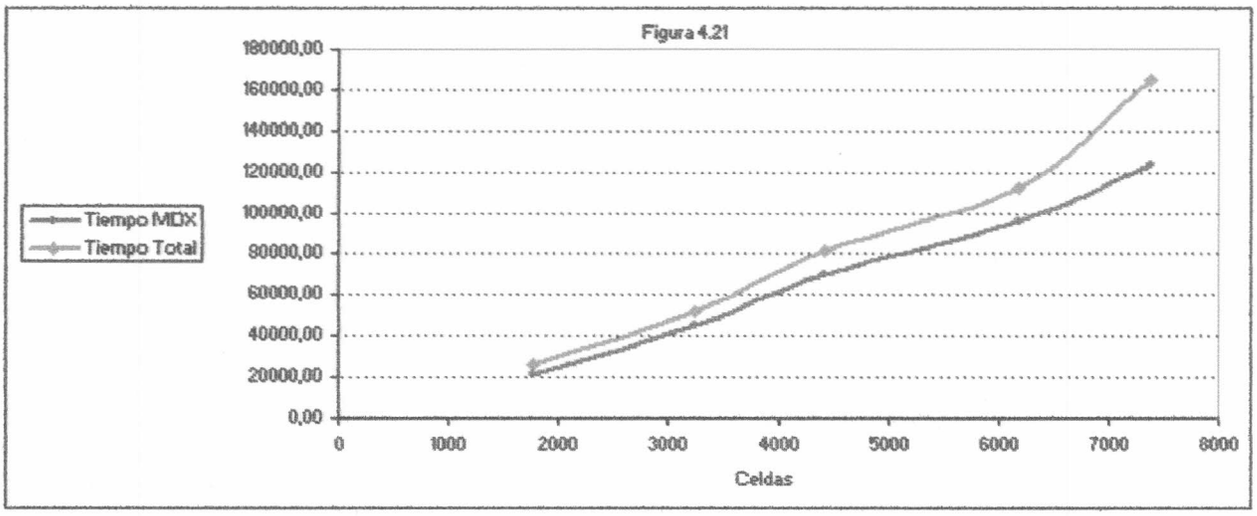
Dimension1-300-Level1 Dimension2-700- Level1 0-Sparse

TabName	PathsRevis.	Paths	Coords	AllCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0012_1	6	300	4	924	0,020	0,020	6969,00	11837,00	8400	132,59
Cubo0012_2	12	300	8	1848	0,040	0,040	15210,00	22112,00	16800	121,50
Cubo0012_3	18	300	11	2541	0,060	0,060	22904,00	30624,00	25200	110,94
Cubo0012_4	24	300	12	2772	0,080	0,080	28452,00	37524,00	30800	97,43
Cubo0012_5	30	300	15	3465	0,100	0,100	34090,00	43854,00	39200	101,64



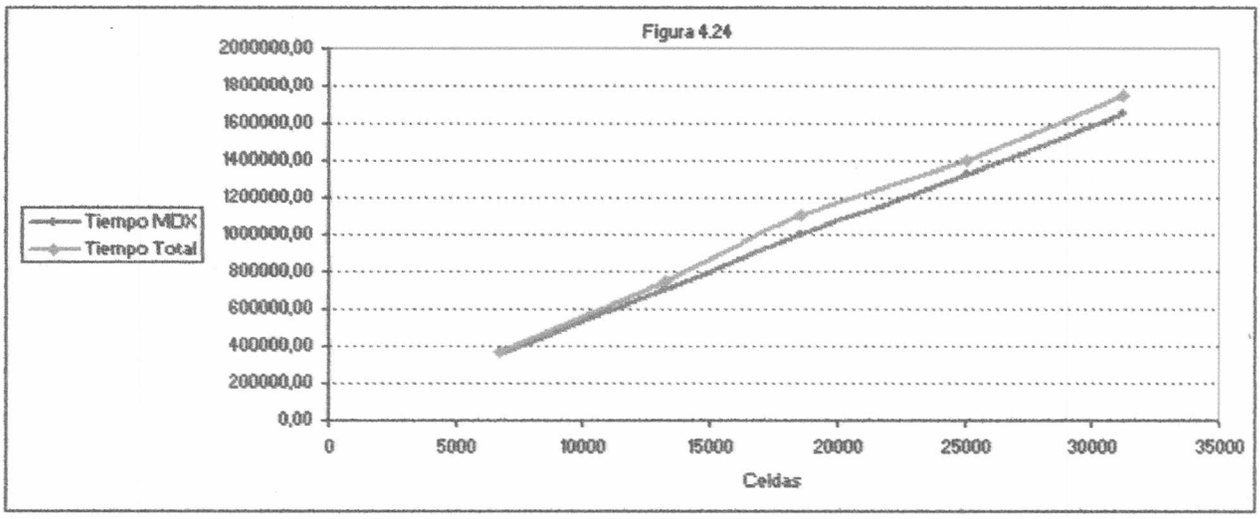
Dimension1-500-Level1 Dimension2-900- Level1 0-Sparse

TabName	PathsRevis.	Paths	Coords	AllCells	p-excep.	l-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0013_1	10	500	6	1770	0,020	0,020	21339,00	26318,00	18000	82,95
Cubo0013_2	21	500	11	3245	0,042	0,042	45185,00	52065,00	34200	71,82
Cubo0013_3	30	500	15	4425	0,060	0,060	70042,00	81247,00	50400	63,18
Cubo0013_4	40	500	21	6195	0,080	0,080	95989,00	111681,00	68400	64,54
Cubo0013_5	50	500	25	7375	0,100	0,100	123048,00	164486,00	86400	59,94



Dimension1-2000-Level1 Dimension2-900- Level1 0-Sparse

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	TSource	Celdas/Seg
Cubo0014_1	44	2000	23	6785	0,022	0,022	355318,00	370694,00	79200	19,10
Cubo0014_2	88	2000	45	13275	0,044	0,044	703575,00	743400,00	156857	18,87
Cubo0014_3	122	2000	63	18585	0,061	0,061	999000,00	1101684,00	219600	18,60
Cubo0014_4	166	2000	85	25075	0,083	0,083	1328975,00	1404200,00	296286	18,87
Cubo0014_5	210	2000	106	31270	0,105	0,105	1657310,00	1751120,00	369486	18,87



Con estos últimos gráficos confirmamos que el comportamiento de estos modelos es coherente, y que a medida que crece la cantidad de celdas afectadas, el tiempo de recálculo de dichas celdas pasa a ser una medida casi directa del tiempo que llevará el procesamiento de la revisión.

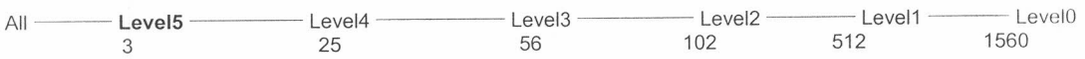
4.8.3 Prueba con un cubo con ocho dimensiones

Hasta el momento las pruebas se hicieron con cubos de dos dimensiones, al principio del capítulo hablamos de que si lo creíamos necesario se harían pruebas con cubos de más dimensiones. Luego de lo visto en el capítulo 4.7, revisando dos dimensiones y viendo que no afectaba al comportamiento de los modelos paramétricos, no se agregaron pruebas con más dimensiones en relación a los modelos. En esta sección probamos otra variable que es el Tiempo, por lo que agregamos una pequeña prueba con un cubo de ocho dimensiones. para ver que tampoco afecta la cantidad de dimensiones al tiempo de recálculo de las celdas afectadas.

Las características de las dimensiones son las siguientes.

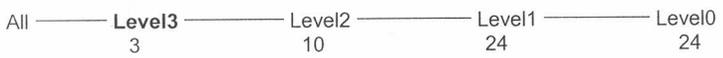
Dimension1

Dimension2



Dimension3

Dimension4



Dimension5

Dimension6

Dimension7

Dimension8

Cubo: SalesX9

Haremos la prueba con una sola revisión, quedara para futuros trabajos extender el estudio bajo estas condiciones.

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	Celdas/Seg
Cubo8Dim	1717	10281	2	17280	0,166	0,077	789033	888508	21,9

Si comparamos con pruebas realizadas en 4.8.2 para la prueba Dimension1-2000-Level1 Dimension2-900- Level1 0-Sparse, que tiene un número similar de cantidad de celdas afectadas vemos que la variable de tiempos se mantiene en el mismo orden.

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	Celdas/Seg
Cubo0014_3	122	2000	63	18585	0,061	0,061	999000,00	1101684,00	18,60

Por lo visto en esta pequeña prueba confirmamos que la cantidad de dimensiones no afecta directamente al tiempo de recálculo de las celdas. Confirma nuestro teoría que esto era así, debido a que la cantidad de dimensiones hace que la cantidad de celdas afectadas sea mayor, ya que la dimensión revisada se cruzará con más dimensiones y producirá mas celdas afectadas, pero no al tiempo de procesamiento de las celdas.

4.8.4 Análisis del tiempo de procesamiento de las celdas

Dados los resultados alentadores obtenidos en los gráficos de la sección 4.8.2, teniendo en cuenta que el tiempo de procesamiento de las celdas es la variable de mayor peso en relación al tiempo, veremos más en detalle como afectan otros factores como el volumen del cubo al tiempo de procesamiento de una celda. Para ello en la tabla que figura a continuación agrupamos los valores obtenidos en los ocho cubos de prueba de tiempos en 4.8.1y 4.8.2, esto lo hacemos para tener un espectro más amplio con relación al tiempo insumido en procesar las celdas.

Celdas/seg	Celdas mater.	Tiempo 1 celda	Procesadores	T ~ 0,01	Dif. Celd/Proc	Increm	10% Celdas	Tiempo
300,00	60000	0,003333333	1	0,003333333			1014,9	10,149
225,00	60000	0,004444444	1	0,004444444			1014,9	10,149
240,00	60000	0,004166667	1	0,004166667			1014,9	10,149
250,00	60000	0,004	1	0,004			1014,9	10,149
257,14	60000	0,003888889	1	0,003888889			1014,9	10,149
305,68	60000	0,003271357	1	0,003271357			1014,9	10,149
266,60	60000	0,003758	1	0,003758			1014,9	10,149
211,36	60000	0,004731156	1	0,004731156			1014,9	10,149
245,86	60000	0,004067	1	0,004067			1014,9	10,149
218,28	60000	0,00458124	1	0,00458124			1014,9	10,149
120,00	210000	0,008333333	1	0,008333333	3,5	0	3488,1	34,881
109,09	210000	0,009166667	1	0,009166667			3488,1	34,881
122,22	210000	0,008181818	1	0,008181818			3488,1	34,881
100,00	210000	0,01	1	0,01			3488,1	34,881
93,75	210000	0,010666667	1	0,010666667			3488,1	34,881
132,59	210000	0,007542208	1	0,007542208			3488,1	34,881
121,50	210000	0,008230519	1	0,008230519			3488,1	34,881
110,94	210000	0,009013774	1	0,009013774			3488,1	34,881
97,43	210000	0,010264069	1	0,010264069			3488,1	34,881
101,64	210000	0,009838384	1	0,009838384			3488,1	34,881
85,31	450000	0,011722222	1	0,011722222	2,142857143	1	7404,5	74,045

58,82	450000	0,017	2	0,0085			7404,5	74,045
63,29	450000	0,0158	2	0,0079			7404,5	74,045
70,71	450000	0,014142857	2	0,007071429			7404,5	74,045
68,74	450000	0,014546667	2	0,007273333			7404,5	74,045
82,95	450000	0,012055932	2	0,006027966			7404,5	74,045
71,82	450000	0,013924499	2	0,00696225			7404,5	74,045
63,18	450000	0,015828701	2	0,00791435			7404,5	74,045
64,54	450000	0,015494592	2	0,007747296			7404,5	74,045
59,94	450000	0,016684475	2	0,008342237			7404,5	74,045
21,07	1800000	0,047463768	5	0,009492754	4	2,5	29529,5	295,295
19,82	1800000	0,050444444	5	0,010088889			29529,5	295,295
20,43	1800000	0,048957672	5	0,009791534			29529,5	295,295
20,30	1800000	0,04925098	5	0,009850196			29529,5	295,295
19,88	1800000	0,050295597	5	0,010059119			29529,5	295,295
19,10	1800000	0,052368165	5	0,010473633			29529,5	295,295
18,87	1800000	0,053	5	0,0106			29529,5	295,295
18,60	1800000	0,053753027	5	0,010750605			29529,5	295,295
18,87	1800000	0,053	5	0,0106			29529,5	295,295
18,87	1800000	0,053	5	0,0106			29529,5	295,295

Tabla 4.8.3

Detalle de las columnas:

Celdas/seg: Cantidad de celdas que se pueden reprocesar en 1 segundo.
Celdas mater.: Cantidad de celdas materializadas del cubo.
Tiempo 1 celda: Tiempo de reprocesamiento de 1 celda.
Procesadores: Estimación de cantidad de procesadores necesarios para que el tiempo de reprocesamiento de una celda para ese caso sea de 0,01 (valor arbitrario considerado aceptable para analizar el comportamiento)
T ~ 0,01: Tiempo exacto de procesamiento de una celda con n procesadores.
Dif. Celd/Proc.: Diferencia de volumen materializado respecto del grupo anterior.
Increm.: Incremento de cantidad de procesadores respecto del grupo anterior, para ver la relación entre el incremento en el volumen y el incremento en la cantidad de procesadores.
10% Celdas: Valor arbitrario de 10% de celdas a recalcular para tomar como cota, para ver como se comportarían los tiempos con n procesadores. Es el 10% de las celdas totales en los TargetLevels de las dimensiones.
Tiempo: Tiempo de procesamiento del 10% de las celdas de cada cubo particular, teniendo en cuenta el agregado de procesadores para conservar un valor constante en el procesamiento de cada celda.

Características del Hardware de testeo

PC Notebook Pentium III 900 Mhz + 128 Mb de RAM + HD IDE de 20 Gb

Software
Microsoft Windows 2000 Server + SP2
Microsoft SQL Server 2000 + SP2
Microsoft Analysis Server 2000 + SP2
Microsoft MDAC 2.6

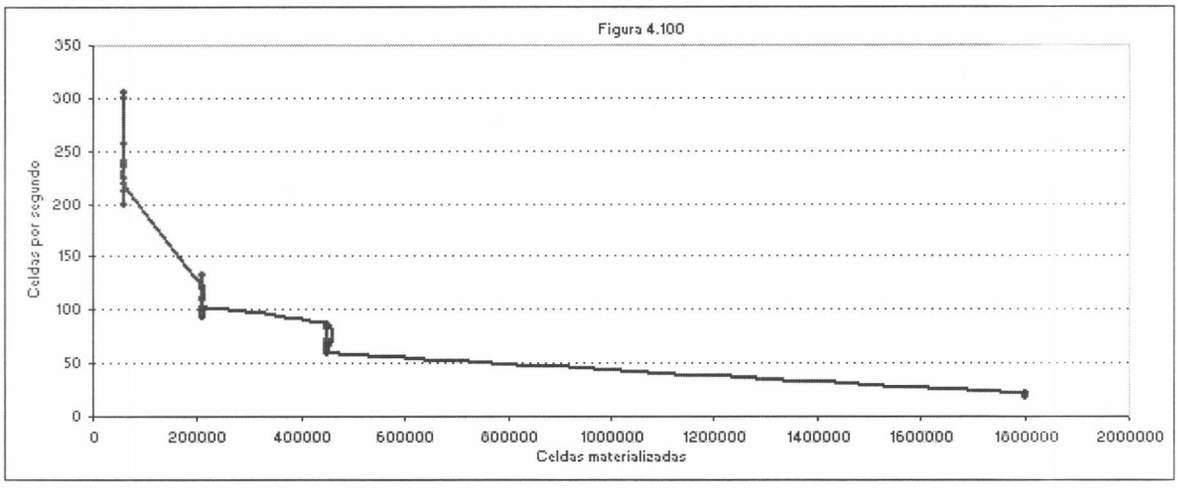
Componente para Revisión Multidimensional de Dimensiones y Cubos (ExceptMD)

Una máquina más adecuada para hacer testeos más reales podría ser un Dual Pentium III 900 Mhz + HD SCSI + 1 Gb de RAM

Celdas procesadas por segundo en relación al Volumen del cubo.

Como vemos en la columna *Tiempo 1 Celda* de la Tabla 4.8.3, el tamaño del cubo afecta al tiempo de recálculo de cada celda, esto se debe a que las consultas MDX tardan más cuando el volumen del cubo es mayor. Obviamente que estos valores dependerán del hardware utilizado para ejecutar los algoritmos. Estos testeos fueron realizados a modo de estudio en una máquina que no podría desempeñarse en la realidad como servidor de un OLAP Server de cubos con un volumen importante. Además en la misma PC corría todo el software necesario para realizar las pruebas. Por eso en el próximo gráfico mostramos como se comporta la variable *Celdas por segundo*, en relación al tamaño del cubo materializado.

Figura 4.100



En el grafico notamos que con el hardware utilizado, el tamaño del cubo influye en el tiempo de procesamiento de las celdas a recalcular, pero la influencia va disminuyendo de forma progresiva a medida que aumenta el tamaño del cubo. Esta caída brusca en la cantidad de celdas por segundo, tiene mucho que ver con el hardware utilizado, ya que no es adecuado para volúmenes importantes. Con cubos más grandes estos valores deberían generarse nuevamente con máquinas adecuadas a los tamaños de los cubos.

Análisis de la cantidad de procesadores

El recálculo de las Celdas Afectadas es claramente un proceso destinado a ser ejecutado en forma paralela, pero debido a que el tamaño del cubo influye sobre el tiempo de procesamiento de las celdas, no podríamos sacar una relación entre el tiempo que tarda en recalcular n cantidad de celdas con un procesador y cuanto tardaría si tengo P procesadores

en paralelo. Teniendo en cuenta que el tiempo de procesamiento de cada celda es similar para todas las celdas afectadas por una revisión particular y una instancia particular, definimos nuevas variables que aparecen en la Tabla 4.8.3.. A continuación tomamos los siguientes valores:

- Tomamos un valor estimado como aceptable para el procesamiento de una celda de 0,01 segundo por celda
- Cantidad de celdas procesadas por segundo con un procesador
- *Tiempo 1 celda*
- Asumimos que si con un procesador proceso n celdas en un segundo, con P procesadores, proceso P x n celdas por segundo.
- *Procesadores,* es la cantidad de procesadores necesarios para mantener un valor constante de procesamiento de 1 celda a medida que crece el volumen del cubo.

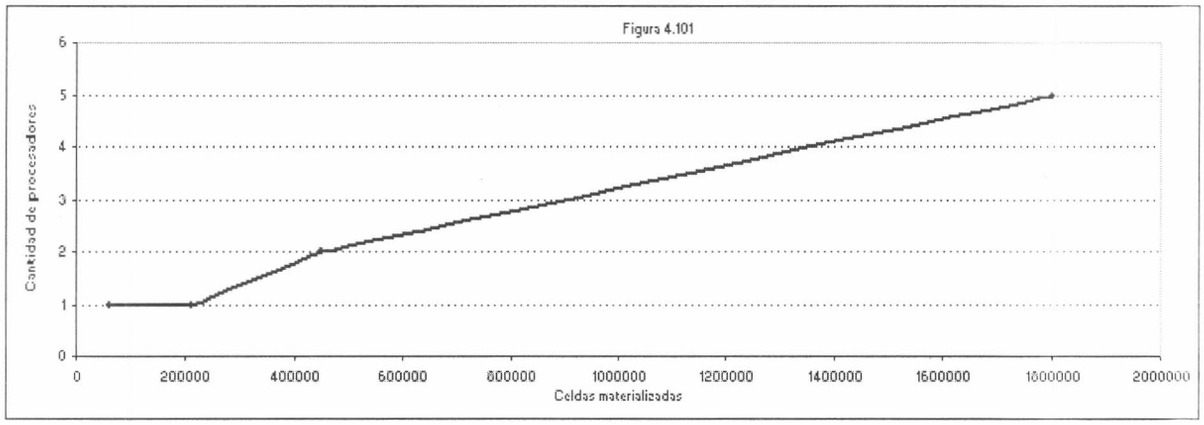
En base a estas variables entonces definimos la variable *T – 0,01,* la que nos da el valor calculado de procesar una celda con n procesadores, donde n aparece en la columna de *Procesadores.*

$$T - 0{,}01 = \frac{\textit{Tiempo 1 Celda}}{\textbf{Procesadores}}$$

Entonces ahora tenemos un valor de tiempo de procesamiento por celda, aproximadamente cercano a 0,01, donde agregamos para cada cubo particular los procesadores necesarios para alcanzar este valor cercano a 0,01 segundos por celda.

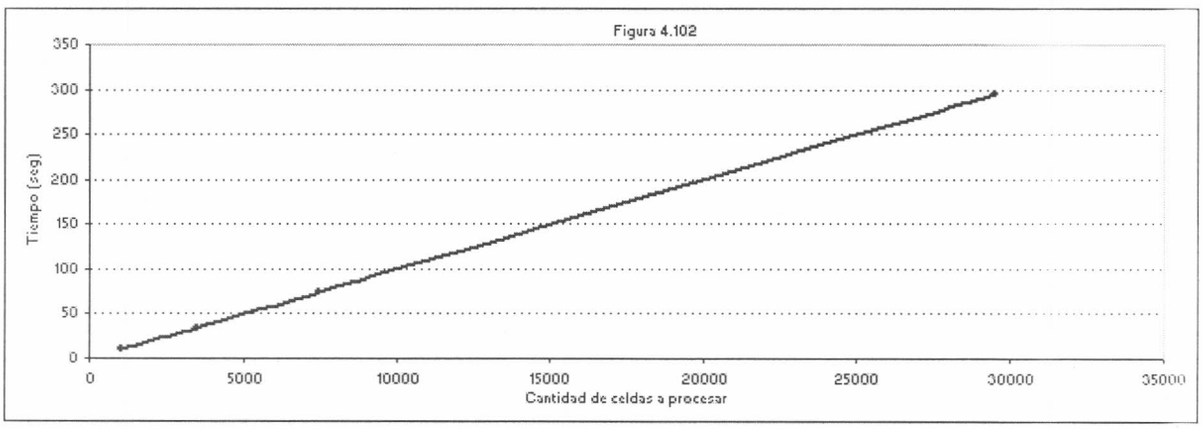
El valor de la columna *10% celdas* se obtiene de tomar el 10% de la cantidad de celdas plausibles de ser revisadas en cada cubo según los TargetLevels seleccionados y a este valor lo multiplicamos por el tiempo insumido en reprocesar una celda (*T – 1 Celda).* Quiere decir que en esta columna tenemos el tiempo en reprocesar el 10% de las celdas de un cubo, en condiciones similares para todos los cubos.

Con esta información, en el próximo gráfico mostramos como se comporta el requerimiento de procesadores según el tamaño del cubo, partiendo de los cubos que hemos testeados, que son de tamaño limitado debido al hardware utilizado para el testeo.



En el gráfico vemos como crece linealmente la necesidad de nuevos procesadores para mantener un valor constante de tiempo de procesamiento de cada celda en aproximadamente 0,01 seg,

Por último vemos como se comportar la curva teniendo en cuenta el procesamiento del *10% de las celdas* y cumpliendo con la premisa del aumento en la cantidad de procesadores para mantener un valor de procesamiento por celda constante. El gráfico relaciona la cantidad de celdas a procesar con el tiempo insumido para procesarlas.



Vemos que la curva tiene un crecimiento lineal en el tiempo de procesamiento a medida que aumenta la cantidad de celdas a procesar.

En base a estos tiempos podemos arriesgar una recomendación de cuando sería aceptable un procesamiento de la revisión on-line y cuando se recomendaría un procesamiento off-line. Si tomamos un valor arbitrario de 60 segundos como lo que un usuario esta dispuesto a esperar para una revisión y considerando que se siguen los lineamientos de que se agregan procesadores según el tamaño del cubo, podríamos decir que manteniendo la cota máxima de 10% de celdas afectadas, podríamos procesar cubos de

hasta (60 / 0,01) * 10 = 60000 celdas en los niveles afectados. Esto es aproximadamente 365.000 de celdas materializadas, este último valor sale de una extrapolación con los valores de la tabla 74 seg -> 450.000, 296 seg -> 1.800.000 => 60 seg ~> 365.000 celdas. Esta estimacion serviría para la máquina en la que estamos trabajando, para tomar la decisión de si el proceso puede ser tratado on-line proponemos una herramienta, que puede desarrollarse como una extensión del componente de revisión, que funcionaría de la siguiente manera:

Debido a que encontramos variables que de forma inexorable afectan al tiempo de revisión de un cubo, como es la cantidad de celdas afectadas, proponemos dotar al usuario de una herramienta que le permita conocer el valor de las variables y una estimación de tiempos de procesamiento. Proponemos esto, ya que es muy complicado predecir cuando algo se puede hacer on-line o no, esto depende mucho del modelo del negocio y del usuario, y luego de haber desarrollado el componente de revisión, notamos que para poder estimar valores que le permitan al usuario decidir que hacer, no se requiere mucho tiempo de procesamiento.

4.9 Análisis del tiempo en relación a las Celdas Source

Las Celdas Source definidas al principio del capítulo, son aquellas que produjeron la pérdida o la ganancia de alguna de las Celdas Afectadas. En los próximos gráficos veremos como afecta a los tiempos de procesamiento de las Celdas Afectadas, la cantidad de Celdas Source relacionada con cada una.

Probamos como afecta al *Tiempo MDX* la cantidad de Celdas Source para una cantidad constante de celdas afectadas. Fijamos la cantidad de Celdas Afectadas a un valor constante para ver directamente como influyen las Celdas Source. Además construimos la dimensión a revisar de manera que el último nivel sea muy numeroso en relación a su padre, de esta manera poder probar rangos grandes de Celdas Source.

Recordamos el formato de las revisiones a aplicar:

Rev0001- Rev0005: r1: Set [Level1] = C1 Where [Level0] >= Cs1 And [Level0] <= Cs1

Dimension1

All	Level3	Level2	**Level1**	Level0
	1	1	6	5000

Dimension2

All	**Level5**	Level4	Level3	Level2	Level1	Level0
	3	23	55	102	199	600

Cubo: Sales5000
Revisiones: Rev0001 -/- Rev0017

Tabéame	PathsRevisados	PathsTotales	AffCells	Source/AffCells	TiempoMDX	TiempoTot	Source	p-excep	I-excep
Cubo0401_5	12	5000	9	1,222	2143	3945	11	0	0
Cubo0401_51	17	5000	9	1,778	3198	6460	16	0	0
Cubo0401_52	28	5000	9	2,889	5248	8182	26	0,1	0,1
Cubo0401_53	34	5000	9	3,444	6408	9675	31	0,1	0,1
Cubo0401_41	45	5000	9	4,556	8011	11126	41	0,1	0,1
Cubo0401_42	89	5000	9	9,000	15813	19298	81	0,2	0,2
Cubo0401_43	134	5000	9	13,444	24235	26779	121	0,3	0,3
Cubo0401_44	178	5000	9	17,889	33869	36072	161	0,4	0,4
Cubo0401_45	223	5000	9	22,333	43643	46507	201	0,4	0,4
Cubo0401_46	267	5000	9	26,889	53437	56351	242	0,5	0,5
Cubo0401_47	311	5000	9	31,222	65635	69009	281	0,6	0,6
Cubo0401_48	356	5000	9	34,667	77972	80826	312	0,7	0,7
Cubo0401_49	400	5000	9	44,444	89898	95126	400	0,8	0,8
Cubo0401_4	442	5000	9	66,667	99093	101306	600	0,9	0,9
Cubo0401_3	664	5000	9	88,889	160972	164908	800	0,13	0,13
Cubo0401_1	1108	5000	9	111,111	259915	266794	1000	0,22	0,22
Cubo0401_6	2219	5000	12	166,667	517453	544973	2000	0,44	0,44

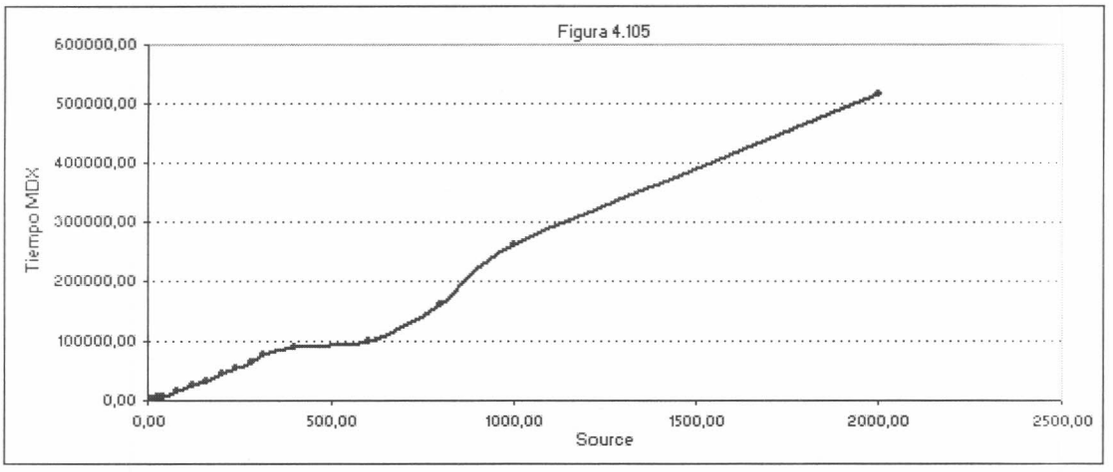
Los valores de la tabla que se tomarán en cuenta para los gráficos son los siguientes:

AffCells: cantidad de celdas afectadas.
Source/AffCells (Psource): es un coeficiente que indica la relación promedio entre la cantidad de Celdas Afectadas y las Celdas Source consideradas. Da una idea de cuantas Celdas Source se tienen en cuenta en el filtro del MDX de recálculo de cada Celda Afectada.
TiempoMDX: tiempo puro de ejecución de consultas MDX para recalcular las celdas afectadas.
TiempoTot: tiempo total de la revisión.
Source: cantidad de Celdas Source consideradas en el MDX. A diferencia de la columna TSource que aparecía en las primeras pruebas del capítulo, que indicaba la cantidad de celdas source consultadas, el valor de Source indica la cantidad elementos de filtro que se incluyeron en la consulta MDX, que es lo que realmente marca la diferencia en el tiempo de procesamiento de los MDX.

4.9.1 Análisis de tiempos

En este gráfico vemos como se comporta el valor de Source, que representa a la cantidad de Celdas Source que se ponen como filtro en las MDX de recálculo, respecto del tiempo de procesamiento de las consultas MDX.

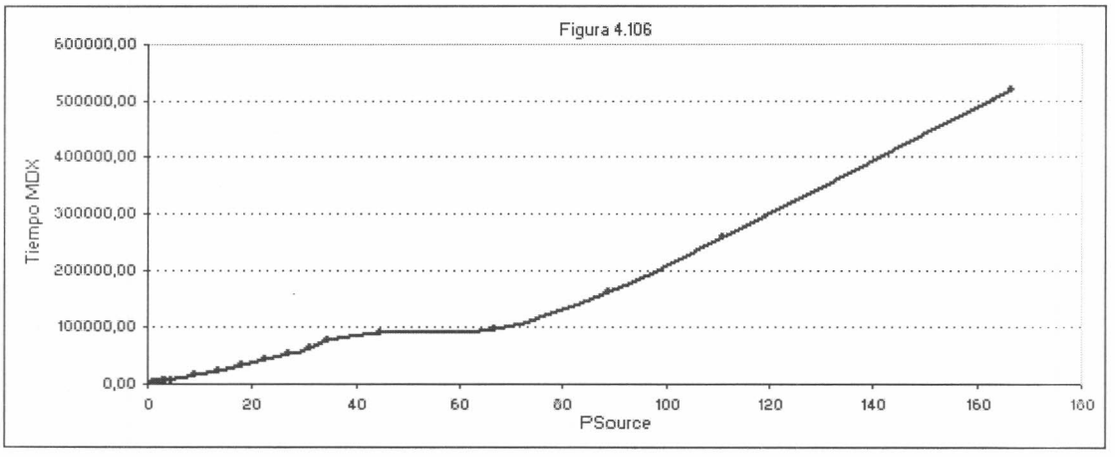
Figura 4.105



Como vemos, el tiempo crece linealmente a medida que aumentamos la cantidad de Celdas Source consideradas como filtro en las consultas MDX utilizadas para recalcular las celdas afectadas.

En el próximo gráfico vemos como se comporta la relación entre las Celdas Afectadas y las Celdas Source (PSource), respecto del tiempo.

Figura 4.106



Vemos que el tiempo también crece linealmente a medida que aumenta la cantidad de Celdas Source por cada Celda Afectada. Hay que tener en cuenta estos tiempos, ya que al crecer linealmente afectarán más al tiempo total de la revisión, cuando los valores de las Celdas Afectadas también crezcan.

4.9.2 Pruebas respecto de las p-excepciones y l-excepciones

A continuación mostramos como se comportan los valores de PSource respecto de las p-excepciones y las l-excepciones, no tomamos cota para estos modelos ya que utilizamos los valores que surgieron de las pruebas de tiempos anteriores.

Figura 4.130

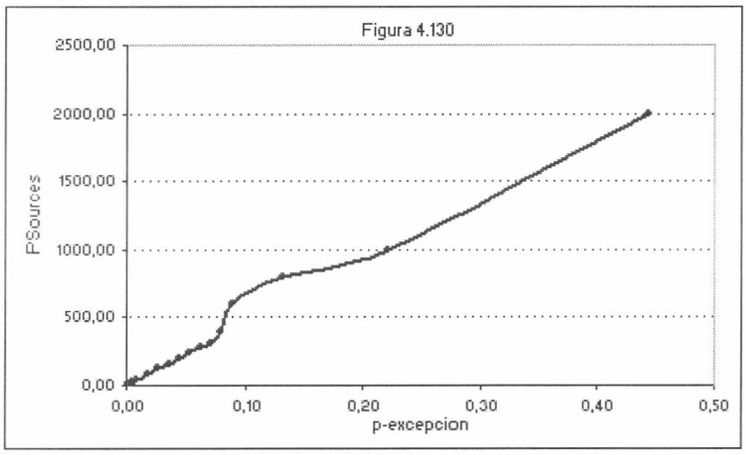
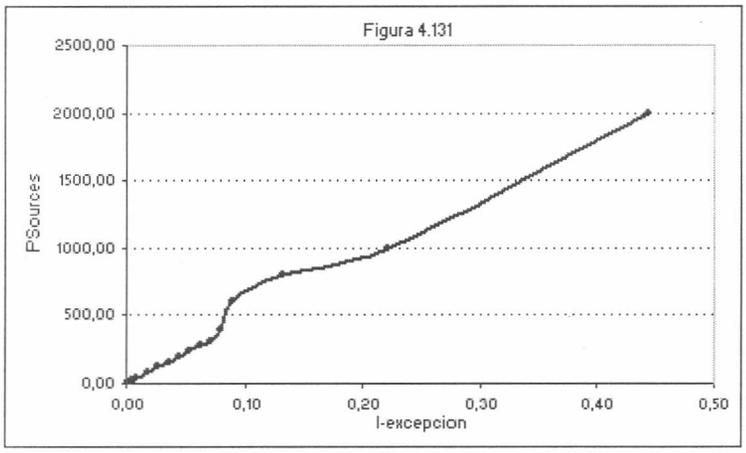


Figura 4.131



Las curvas mantienen un comportamiento creciente respecto de los dos modelos.

4.9.3 Pruebas relacionando las Celdas Source por cada Celda afectada

Definimos CeldasT como el producto entre la cantidad de Celdas Afectadas y la cantidad de Celdas Source.

CeldasT = Celdas Afectadas * Celdas Source

Esta fórmula se definió como un producto, ya que por los estudios anteriores vimos que el Tiempo de procesamiento crece de manera lineal respecto de la cantidad de Celdas Afectadas y que el Tiempo también es de crecimiento lineal respecto de las Celdas Source.

Para ver como se comporta CeldasT respecto del tiempo, tomaremos cuatro de los ocho cubos testeados en el punto 4.8, y veremos el comportamiento de las curvas.

Dimension1-100-Level1 Dimension2-600- Level5 0-Sparse

Dimension1

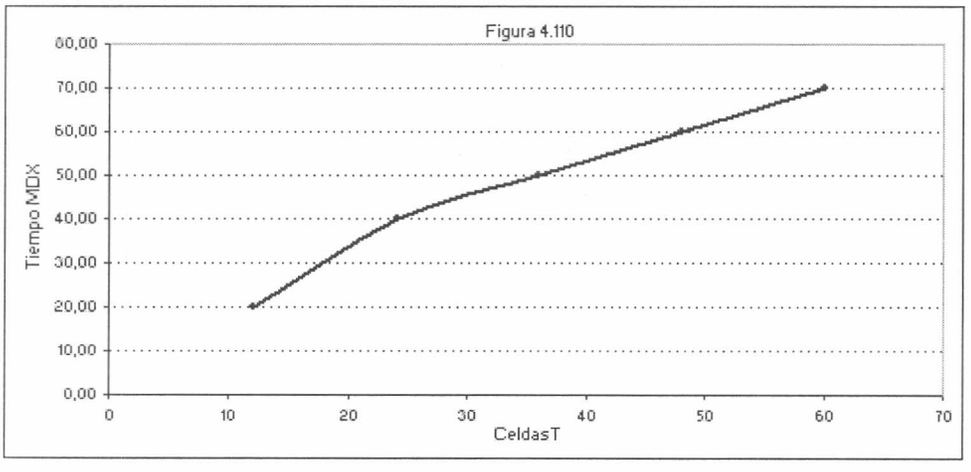
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
13 26 51 100

Dimension2

All ——— **Level5** ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0
3 23 55 102 199 600

TabName's: Cubo0001_1 - Cubo0001_2 - Cubo0001_3 - Cubo0001_4 - Cubo0001_5
Cubo: Sales100_600_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	Source	CeldasT
Cubo0001_1	2	100	2	6	0,020	0,020	20,00	1923,00	2	12
Cubo0001_2	4	100	3	9	0,040	0,040	40,00	1913,00	2,67	24,03
Cubo0001_3	6	100	4	12	0,060	0,060	50,00	2083,00	3	36
Cubo0001_4	8	100	5	15	0,080	0,080	60,00	1763,00	3,2	48
Cubo0001_5	10	100	6	18	0,100	0,100	70,00	1872,00	3,34	60,12



Dimension1-2000-Level1 Dimension2-900- Level5 0-Sparse

Dimension1

All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
251 501 1001 2000

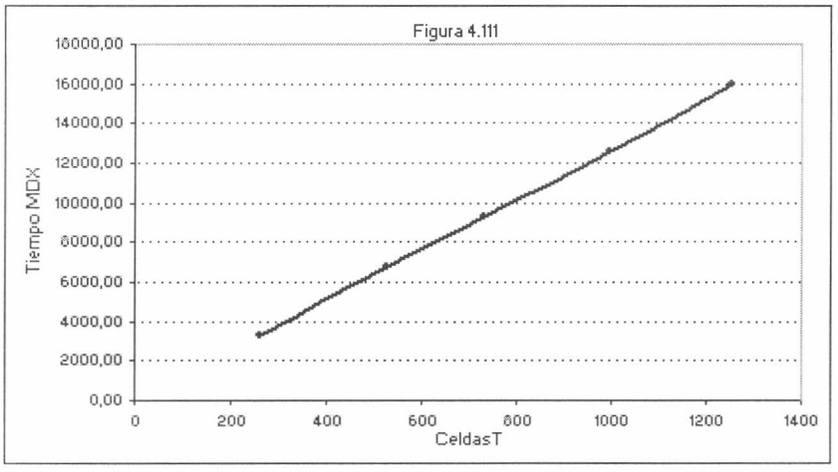
Dimension2

All ——— **Level5** ——— Level4 ——— Level3 ——— Level2 ——— Level1 ——— Level0
3 23 55 102 295 900

TabName's: Cubo0004_1 - Cubo0004_2 - Cubo0004_3 - Cubo0004_4 - Cubo0004_5
Cubo: Sales2000_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	Source	CeldasT
Cubo0004_1	44	2000	23	69	0,022	0,022	3275,00	6129,00	3,8	262,2
Cubo0004_2	88	2000	45	135	0,044	0,044	6810,00	11817,00	3,91	527,85
Cubo0004_3	122	2000	63	189	0,061	0,061	9253,00	15792,00	3,87	731,43
Cubo0004_4	166	2000	85	255	0,083	0,083	12559,00	18567,00	3,90	994,5
Cubo0004_5	210	2000	106	318	0,105	0,105	15994,00	23314,00	3,94	1251,92

Figura 4.111



Dimension1-100-Level1 Dimension2-600- Level1 0-Sparse

Dimension1

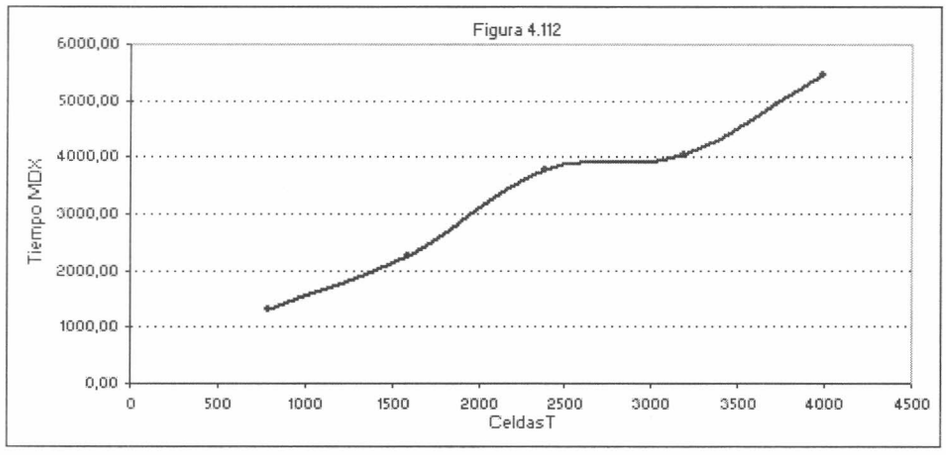
All — Level3 — Level2 — **Level1** — Level0
13 26 51 100

Dimension2

All — Level5 — Level4 — Level3 — Level2 — **Level1** — Level0
3 23 55 102 199 600

TabName's: Cubo0011_1 - Cubo0011_2 - Cubo0011_3 - Cubo0011_4 - Cubo0011_5
Cubo: Sales100_600_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	l-excep.	T-MDX (1)	T-Total	Source	CeldasT
Cubo0011_1	2	100	2	398	0,020	0,020	1302,00	4706,00	2	796
Cubo0011_2	4	100	3	597	0,040	0,040	2244,00	6269,00	2,67	1593,99
Cubo0011_3	6	100	4	796	0,060	0,060	3766,00	8492,00	3	2388
Cubo0011_4	8	100	5	995	0,080	0,080	4047,00	8142,00	3,2	3184
Cubo0011_5	10	100	6	1194	0,100	0,100	5470,00	10596,00	3,34	3987,96



Dimension1-2000-Level1 Dimension2-900- Level1 0-Sparse

Dimension1

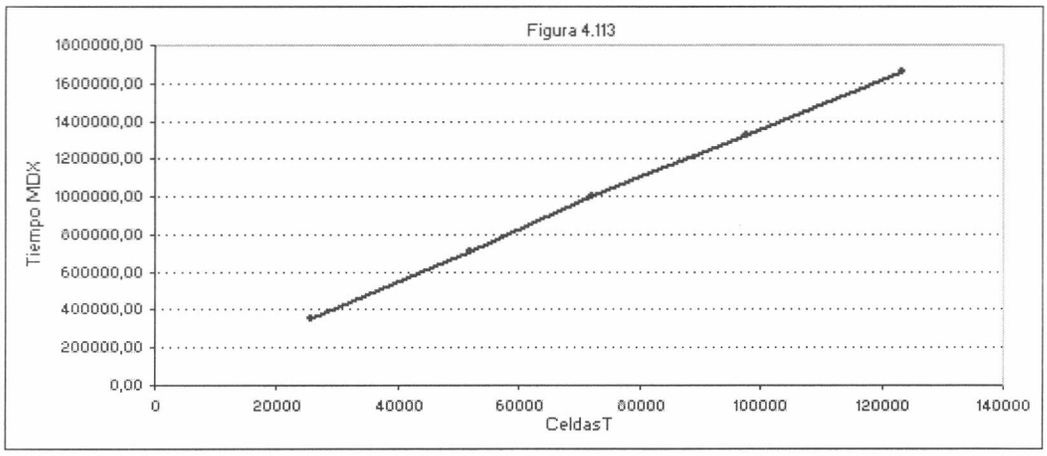
All ——— Level3 ——— Level2 ——— **Level1** ——— Level0
251 501 1001 2000

Dimension2

All ——— Level5 ——— Level4 ——— Level3 ——— Level2 ——— **Level1** ——— Level0
3 23 55 102 295 900

TabName's: Cubo0014_1 - Cubo0014_2 - Cubo0014_3 - Cubo0014_4 - Cubo0014_5
Cubo: Sales2000_900_0
Revisiones: Rev0010 - Rev0014

TabName	PathsRevis.	Paths	Coords	AffCells	p-excep.	I-excep.	T-MDX (1)	T-Total	Source	CeldasT
Cubo0014_1	44	2000	23	6785	0,022	0,022	355318,00	370694,00	3,80	25783
Cubo0014_2	88	2000	45	13275	0,044	0,044	703575,00	743400,00	3,91	51905,25
Cubo0014_3	122	2000	63	18585	0,061	0,061	999000,00	1101684,00	3,87	71923,95
Cubo0014_4	166	2000	85	25075	0,083	0,083	1328975,00	1404200,00	3,90	97792,5
Cubo0014_5	210	2000	106	31270	0,105	0,105	1657310,00	1751120,00	3,94	123203,8



Debido a que los valores involucrados en la fórmula CeldasT son de crecimiento lineal respecto del tiempo, el tiempo también crece linealmente a medida que crece la relación CeldasT. Todo lo que tiene que ver con tiempos son medidas dependientes del entorno de análisis, como ser el hardware utilizado y el volumen de los casos de prueba.

Lo visto en esta sección no invalida de ninguna manera las pruebas de tiempos realizadas en la sección 4.8, ya que el valor de las Celdas Source es un valor asociado directamente con cada Celdas Afectada a recalcular. Los gráficos de la sección 4.8 sirven como muestra de cómo progresan las curvas, y están realizadas para un determinado factor de Celdas Source. Si afectáramos las pruebas con mayor o menor cantidad de Celdas Sources por celda afectada, las curvas no variarán significativamente su forma, ya que ambas variables son de crecimiento lineal en el tiempo.

Una opción que podría brindar el componente desarrollado, es la posibilidad de que la aplicación informe la cantidad de Celdas Afectadas que va a procesar, antes de hacerlo y también informar la Cantidad de Celdas Source, basándose en esta información podemos estimar si conviene ejecutar el proceso en el momento o dejarlo para una ejecución Off-Line. También serviría para indicar si hay algún problema en la definición de la excepcion, debido al volumen de Celdas a procesar. Para obtener estos valores el tiempo insumido está en el orden de 1/50 parte del tiempo total, este valor surge de aproximadamente 300 cubos procesados.

4.10 Síntesis

Hemos estudiado el comportamiento del algoritmo de revisión, definiendo distintos modelos, buscando que abarquen la mayoría de los parámetros involucrados en la definición de una revisión con sus reglas y de la estructura de las instancias de las dimensiones revisadas.

Luego basándonos en esos modelos, ejecutamos pruebas sobre distintos cubos, variando la cardinalidad de los niveles de las dimensiones, variando los TargetLevels de las revisiones, para así variar la cantidad de celdas afectadas por la revisión. También variamos la cantidad de reglas por revisión, afectando de esta manera a más de un nivel dentro de la jerarquía, considerando factores como la relación de cardinalidad entre los niveles (FIO). Por último variamos la cantidad de dimensiones revisadas. En las últimas secciones hicimos pruebas respecto del tiempo de procesamiento, y confirmamos que los valores de celdas afectdas y celdas source tienen una relación directa con el tiempo insumido en recalcular las celdas afectadas. Luego de hacer todas estas pruebas, podemos concluir que existe un comportamiento predecible, y de crecimiento lineal, dentro de las cotas establecidas, donde es aceptable el modelo de definición de excepciones.

Capítulo 5 Conclusiones generales

Hemos desarrollado los algoritmos propuestos en [MV01] y [MV02], adaptándolos a las herramientas utilizadas, ya que los mismos no habían sido implementados antes, incluyendolos dentro de un componente que corre del lado del servidor OLAP.

Hemos presentado y motivado los beneficios de la definición de excepciones, a partir de eso, hemos estudiado el comportamiento de los algoritmos de revisión, definiendo distintos modelos paramétricos. Los resultados arrojados por el algoritmo, fueron obtenidos luego de variar las siguientes condiciones de entorno: volumen de los cubos a procesar, cantidad de miembros de los niveles de las dimensiones, cantidad de dimensiones a revisar, cantidad de dimensiones del cubo, N-Sparse del cubo, cantidad de reglas definidas en las revisiones y cantidad de celdas afectadas. Partiendo de estas pruebas y haciendo un análisis del comportamiento de distintas variables y modelos concluimos que: basados en los modelos paramétricos definidos, existe un comportamiento predecible del algoritmo y de las variables Celdas Afectadas y Celdas Source, que afectan de manera significativa a la viabilidad de una revisión.

En relación a los tiempos de procesamiento, tanto las curvas de tiempo de procesamiento de las celdas (Tiempo-MDX), como del tiempo total insumido en la revisión (T-Total), son de comportamiento coherente y de crecimiento lineal en relación al incremento de celdas a procesar. Los resultados obtenidos nos dan parámetros como para confirmar que la propuesta de un método de revisión de cubos de datos con excepciones es viable y funcionalmente útil para dar mayor potencia a las herramientas de OLAP existentes hoy en día.

5.1 Contribuciones

Implementando el componente de revisión, llevamos a la práctica algoritmos que estaban definidos teóricamente. Con esta implementación y con los análisis realizados confirmamos que es factible realizar revisiones de cubos multidimensionales dentro de entornos y parámetros determinados.

Luego del análisis realizado, detectamos variables que de forma inexorable afectarán al tiempo de revisión de un cubo. Proponemos dotar al usuario de una herramienta que le permita conocer el valor de estas variables, que básicamente son las Celdas Afectadas y Celdas Source, y una estimación de tiempos de procesamiento. Nuestra propuesta se motiva en que predecir cuando un proceso se puede ejecutar on-line o no, depende mucho del modelo del negocio y del usuario involucrado. Luego de haber desarrollado el componente de revisión notamos que, con un tiempo de procesamiento despreciable en comparación al tiempo total que podría insumir la revisión, podemos estimar los valores que permitirán al usuario tomar una decisión. Esto ayudará a determinar donde es aceptable el modelo de definición de excepciones.

5.2 Futuros trabajos

En relación al desarrollo del componente de revisión, proponemos trabajar más en la optimización del acceso a los miembros de los niveles de las dimensiones, para niveles superiores al 0 y al 1.

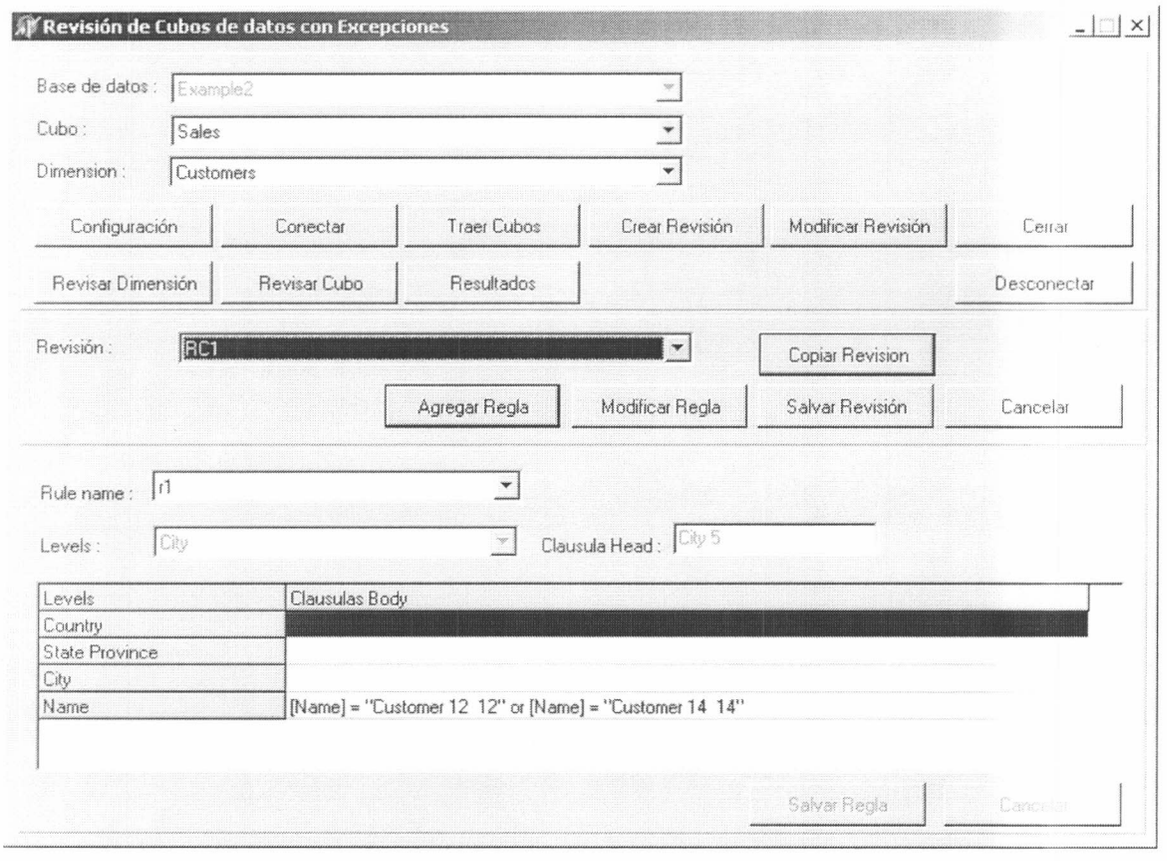
En cuanto a futuros desarrollos, sería importante adaptar el algoritmo de recálculo de las celdas afectadas por la revisión, para que pueda ejecutarse en un ambiente paralelo. De esta manera tendremos la posibilidad de reducir el tiempo de recálculo de manera importante.

Otra propuesta seria trabajar en la extensión de los algoritmos de revisión, para poder ser utilizados en otros servidores de OLAP, que no sean SQL Server. Por último extender el componente para que basándose en el Hardware, las Celdas Afectadas y las Celdas Source, brinde estimaciones de tiempos de revisión.

Apéndice A Manual del usuario de la aplicación

En este capítulo mostramos las pantallas de la interfaz de cliente que servirán para definir revisiones para los distintos cubos y dimensiones, ejecutar los algoritmos de revisiones de dimensiones y de cubos, consultar datos del metadata del Análisis Server y ver algunos de los resultados arrojados por los algoritmos de revisión.

A.1 Formulario Revisión de cubos de datos con Excepciones.



Es el formulario principal del sistema, incluimos la mayor cantidad de información y funcionalidad para facilitar el uso.

Panel superior

Combo **Base de datos**: aquí se selecciona lo que en Análisis Manager se llama Database, es el contenedor de los Cubos y las dimensiones.

Combo **Cubo**: as el cubo a revisar dentro de la base de datos.

Combo **Dimension**: se selecciona una dimensión dentro del cubo seleccionado, para hacer una revisión sobre esa dimensión o para dar de alta o modificar las revisiones.

Botón **Configuración**: llama al formulario de configuración del sistema.

Botón **Conectar**: es el primer botón que se habilita, para conectarse a la base de datos del Análisis Server.

Botón **Traer Cubos**: busca los cubo procesados dentro de la base de datos seleccionada.

Botón **Crear Revisión**: permite crear una nueva revisión para el cubo y la dimensión seleccionadas.

Botón **Modificar Revisión**: permite modificar una revisión para el cubo y la dimensión seleccionadas.

Botón **Cerrar**: cierra la aplicación.

Botón **Revisar Dimensión**: abre el formulario para parametrizar y ejecutar la revisión de la dimensión seleccionada.

Botón **Revisar Cubo**: abre el formulario para parametrizar y ejecutar la revisión del cubo seleccionado.

Botón **Resultados**: abre un formulario donde se permite seleccionar un identificador de Cubo revisado para ver los resultados que se obtuvieron en la última revisión.

Botón **Desconectar**: permite desconectarse de la base de datos, para conectarse a otra.

Panel medio

Combo **Revisión**: en el alta se convierte a tipo texto para permitir el ingreso del nombre de la nueva revisión, en modificación es un combo que permite seleccionar la revisión a modificar.

Botón **Agregar Regla**: permite dar de alta una nueva regla para la revisión activa.

Botón **Modificar Regla**: permite modificar una regla para la revisión activa.

Botón **Salvar Revisión**: graba en la base de datos la revisión activa. Siempre que se de alta o modifique una revisión deberá presionarse este botón para que quede almacenado en la base de datos.

Botón **Copiar Revisión**: permite copiar una revisión del cubo y dimensión actual a otro cubo o dimensión.

Panel inferior

Combo **Regla**: en el alta se convierte a tipo texto para permitir el ingreso del nombre de la nueva revisión, en modificación es un combo que permite seleccionar la revisión a modificar.

Combo **Nivel**: permite seleccionar el nivel donde se aplicará la cláusula SET.

Texto **Cláusula Head**: aquí se ingresa el valor a asignar al nivel si se cumple la regla. El valor se ingresa como un string, sin comillas ni caracteres especiales.

Grilla columna **Niveles**: indica los niveles de la dimensión.

Grilla columna **Cláusula Body**: permite el ingreso de la condición WHERE para el nivel activo. Cada término de la cláusula tiene que tener el siguiente formato:

[Nombre del nivel] {Operador} "Valor"

por ejemplo: [Name] = "Brenda Bloomberg"

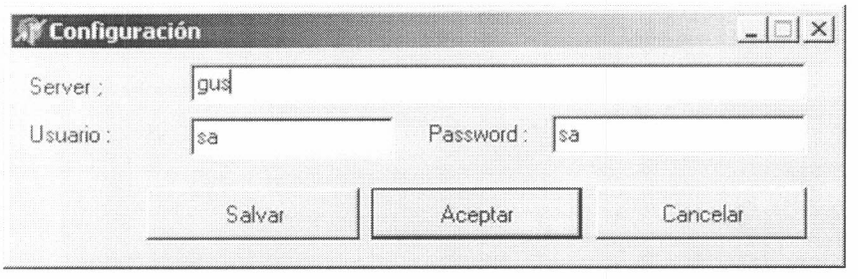
El Nombre del nivel debe ir entre corchetes, el Valor siempre es un string entre comillas y la gramática está definida en el documento principal (ver Parser).

Botón **Salvar Regla**: guarda en memoria la regla y permite el ingreso de una nueva regla, para salvar la revisión definitivamente, se debe presionar el botón **Salvar Revisión** del Panel medio. Siempre que se de alta o modifique una regla deberá presionarse este botón.

A.2 Formulario Member Properties

A este formulario se accede dando doble click del mouse en la columna **Cláusula Body** de la Grilla del panel inferior del formulario principal. Muestra los Member Properties del nivel activo, si es que tiene alguna propiedad asociada.

A.3 Formulario Configuración



Este formulario permite la configuración de conexión con el servidor SQL, asume que el Servidor SQL Server tiene los mismos datos de conexión que el Analysis Server.

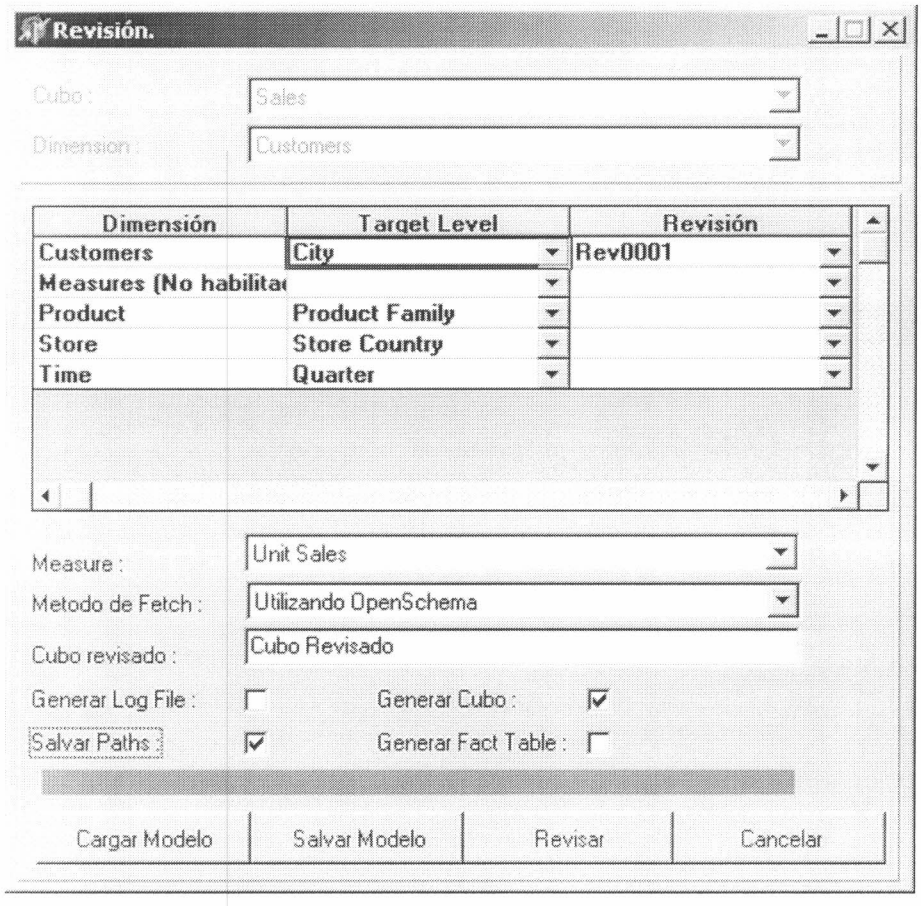
Texto **Server**: nombre del server a conectarse, sin \\.

Texto **Usuario**: nombre del usuario administrador del servidor.

Texto **Password**: password del usuario administrador del servidor.

Botón **Salvar:** graba en dFijos.Ini del directorio de instalación, la información de conexión con el servidor.

A.4 Formulario Revision (Cubos)



Este formulario permite parametrizar la revisión de un cubo.

Panel superior

Combo **Cubo**: no se permite modificar, tiene la misma información del cubo seleccionado en el formulario principal.

Combo **Dimensión**: no se permite modificar, tiene la misma información de la dimensión seleccionada en el formulario principal.

Panel inferior

Grilla columna **Dimensión**: es una columna fija y posee el nombre de todas las dimensiones asociadas al cubo a revisar.

Grilla columna **Target Level**: aquí se selecciona para cada dimensión el Target Level.

Grilla columna **Revisión**: aquí se selecciona la revisión a aplicar sobre la dimensión correspondiente. Puede permanecer en blanco, ya que en general no se revisarán todas las dimensiones al mismo tiempo.

Combo **Measure**: aquí se selecciona el Measure sobre el cual se aplicará la revisión.

Combo **Método de Fetch**: es el método con que se accede a la información del metadata de las dimensiones. Aquí hay dos opciones, se dejó porque fue motivo de la evolución de la aplicación, al principio se comenzó a trabajar con el método "Utilizando ICatalog", pero se notó una deficiencia en la performance, se optimizó con el uso del método "Utilizando OpenSchema", el cual es el recomendado.

Texto **Cubo revisado**: aquí se ingresa un nombre que servirá como identificador de todos los datos asociados a la revisión que se almacenen en la base de datos. Este identificador lo utilizaremos más adelante cuando queramos acceder al formulario de resultados.

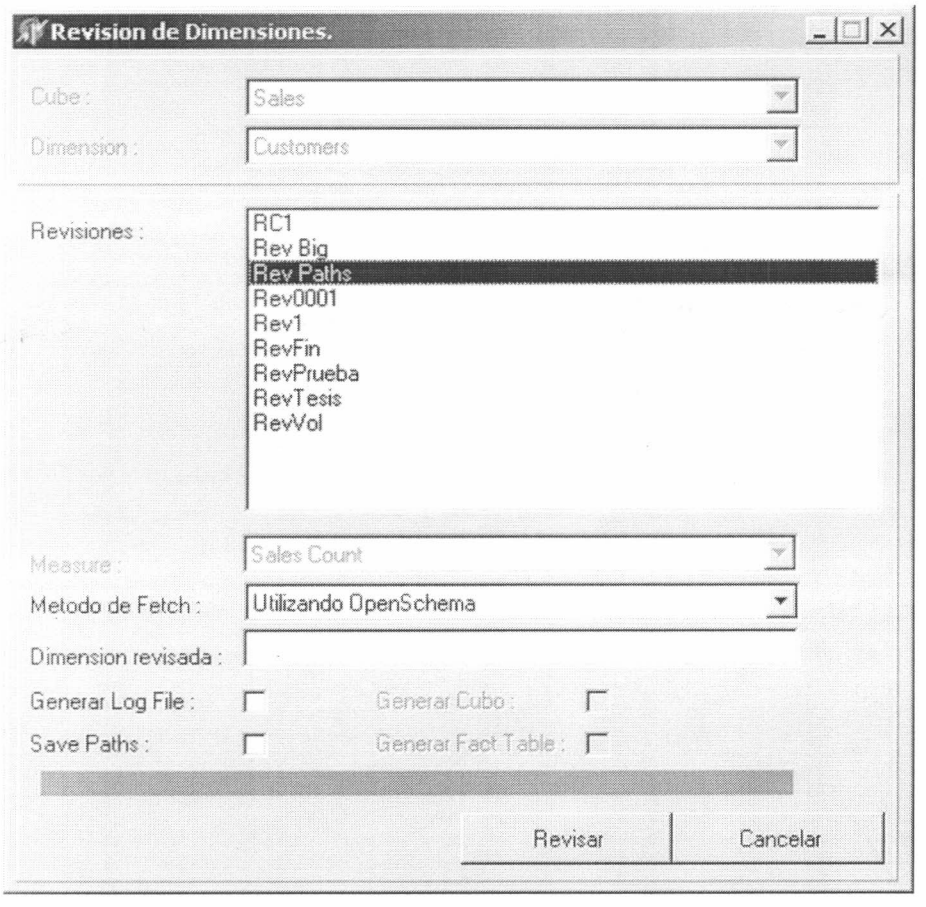
Check **Generar Log File**: habilita la generación de un archivo de log de texto, es con fines de debugging.

Check **Generar Cubo**: al finalizar la revisión genera un nuevo cubo con todas las dimensiones, con nivel inferior en el target level seleccionado y con los miembros y measures resultado de la revisión.

Check **Salvar Paths**: va almacenando en la base de datos los Paths afectados por la revisión.

Check **Generar Fact Table**: esta opción se marca automáticamente si se marca la opción Generar Cubo, ya que genera la Fact Table que se utilizará para la generación del cubo revisado.

A.5 Formulario Revisión (Dimensiones)



Este formulario permite parametrizar la revisión de una dimensión.

Panel superior

Combo **Cubo**: no se permite modificar, tiene la misma información del cubo seleccionado en el formulario principal.

Combo **Dimensión**: no se permite modificar, tiene la misma información de la dimensión seleccionada en el formulario principal.

Panel inferior

Revisiones: es la lista de revisiones que se pueden aplicar sobre la dimension seleccionada.

Combo **Measure**: aquí se selecciona el Measure sobre el cual se aplicara la revisión.

Combo **Método de Fetch**: es el método con que se accede a la información del metadata de las dimensiones. Aquí hay dos opciones, se dejó porque fue motivo de la evolución de la aplicación, al principio se comenzó a trabajar con el método XXX, pero se notó una deficiencia en la performance, se optimizó con el uso del método YYY, el cual es el recomendado.

Texto **Dimensión revisada**: aquí se ingresa un nombre que servirá como identificador de todos los datos asociados a la revisión que se almacenen en la base de datos.

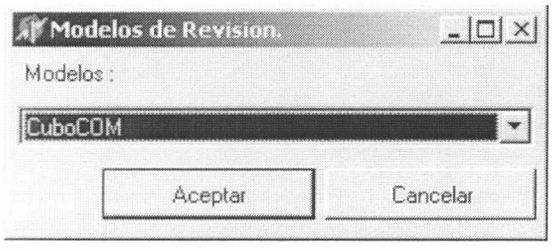
Check **Generar Log File**: habilita la generación de un archivo de log de texto, es con fines de debugging.

Check **Generar Cubo**: no esta habilitado.

Check **Salvar Paths**: va almacenando en la base de datos los Paths afectados por la revisión.

Check **Generar Fact Table**: no esta habilitado.

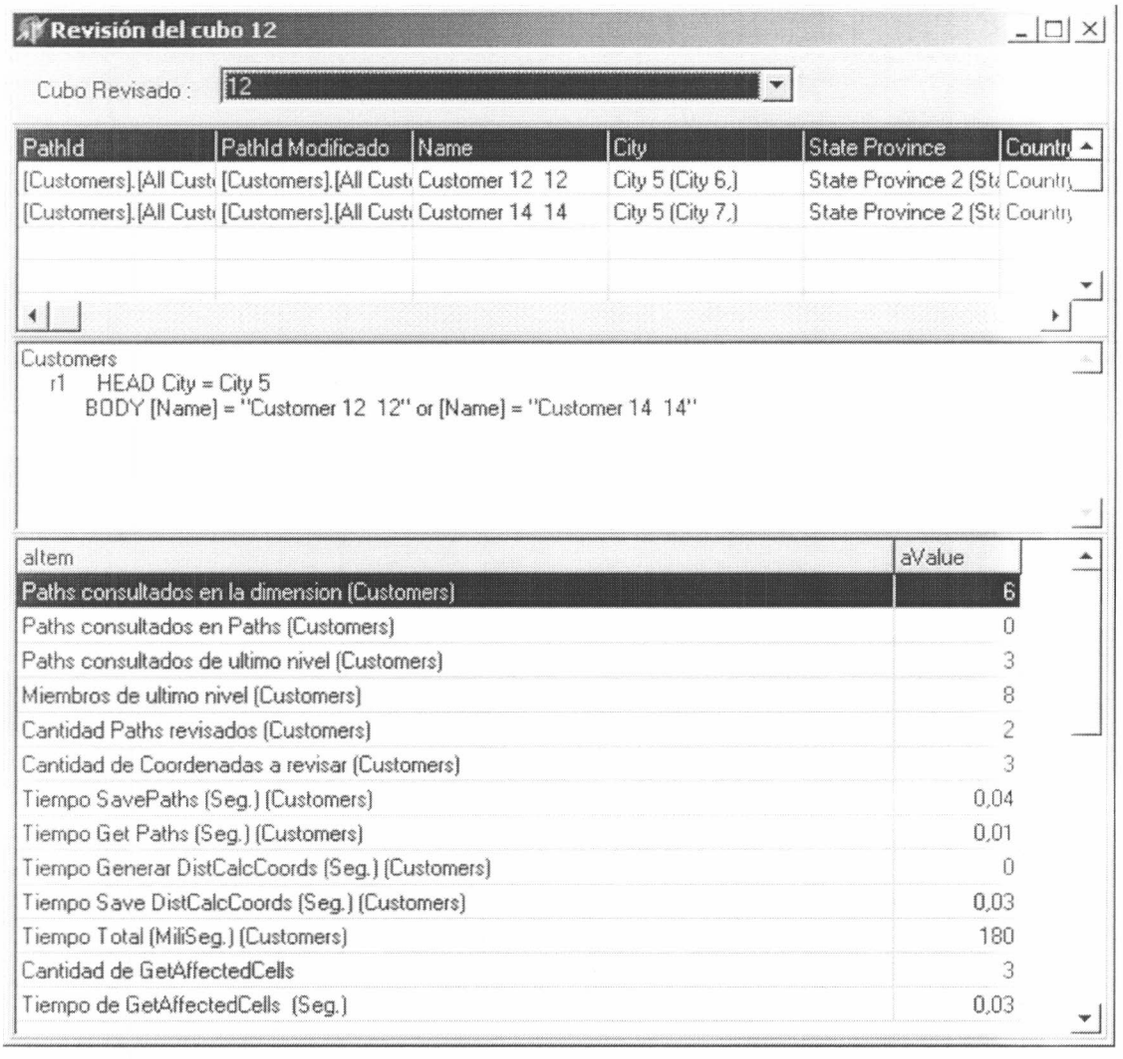
A.6 Formulario Modelos de Revisión



Este formulario es para seleccionar un Modelo de revisión ya almacenado.

Combo **Modelos**: nombre del Cubo revisado con el que se almacenó el modelo.

A.7 Formulario de Resultados



Este formulario permite seleccionar algún identificador de cubo ya revisado para ver los resultados que se obtuvieron en la última revisión.

Combo **Cubo Revisado**: identificador del cubo revisado.

Datos en la grilla superior

En esta grilla aparecen a modo de guía los Paths revisados, por el momento, si se revisa más de una dimensión los paths pueden aparecer mezclados.

Cuando se revise una dimensión por ejemplo Customers, aparecerán los siguientes datos en las columnas:

PathId : [Customers].[All Customers].[Country 1].[State Province 3].[City 6].[Customer 12 12]

PathId Modificado: [Customers].[All Customers].[Country 1].[State Province 2].[City 5].[Customer 12 12]

Name: Customer 12 12

City: City 5 (City 6)

State Province: State Province 2 (State Province 3)

Country: Country 1 (Country 1)

Los datos que aparecen entre paréntesis indican que ese nivel fue modificado en ese path, y entre paréntesis aparecen los valores históricos separados por coma.

En el caso que se revise un cubo, para simplificar, no se puso el nombre de los niveles, ya que cuando se revisa más de una dimensión quedarían mal alineados.

Datos en el memo

Aquí aparecerá una descripción de las reglas de las revisiones aplicadas sobre todas las dimensiones.

Datos en la grilla de resultados

Por dimensión revisada:

Al lado de cada ítem, entre paréntesis aparecerá el nombre de la dimensión.

Paths consultados en la dimensión: paths consultados sobre la instancia de la dimensión (metadata).
Paths consultados en Paths: paths que accedió el algoritmo sobre la estructura de Paths recuperados en memoria.
Paths consultados de ultimo nivel: paths consultados donde se llegó al último nivel (más bajo).
Cantidad de miembros de ultimo nivel: cantidad de miembros del último nivel.
Cantidad de Paths revisados: la cantidad final de paths que fueron revisados.
Cantidad de Coordenadas a revisar: la cantidad de coordenadas que producirán celdas para recalcular.
Tiempo SavePaths: tiempo que demora el almacenamiento optativo de los paths revisados en la base de datos.
Tiempo Get Paths: tiempo que llevó cargar los paths revisados en memoria.

Tiempo Generar DistCalcCoords: tiempo en generar las coordenadas afectadas por la revisión.
Tiempo Save DistCalcCoords: tiempo almacenamiento en la base de datos de la información de las coordenadas seleccionadas.
Tiempo Total: tiempo total de la revisión para la dimensión actual.

Datos Generales

Cantidad de GetAffectedCells: cantidad total de Celdas afectadas por la revisión.
Tiempo de GetAffectedCells: tiempo insumido en el cálculo de las celdas afectadas.
Update de Celdas por segundo: cantidad promedio de celdas por segundo.
Tiempo de Update de Celdas: tiempo insumido en el recálculo de las celdas afectadas.
Tiempo de Creación del Cubo resultado: tiempo de creación del cubo resultado.
Tiempo de Procesamiento del Cubo: tiempo de Procesamiento del Cubo resultado.
Tiempo puro de MDXs de recálculo: tiempo exacto de todos los MDX's ejecutados para el recálculo de las celdas.
Cantidad de Source Cells: cantidad de celdas source consultadas para el recálculo de las celdas afectadas.
Tiempo Total Revisión Cubo: tiempo total para la revisión del cubo.

A.8 Otros datos estadísticos

Como información adicional, en la base de datos relacional OlapDB se guardarán dos tablas conteniendo información muy útil para hacer el análisis, estas tablas son MDTestHead y MDTestDet.

MDTestHead contiene la siguiente información:

TabName: identificador del test
CubeName: nombre del cubo
DimName: nombre de la dimensión
RevId: identificador de la revisión
RevName: nombre de la revisión
PathsRevisados: cantidad de paths revisados
PathsTotales: paths totales de la instancia
Coordenadas: cantidad de coordenadas a revisar.
Levels: cantidad de niveles de la dimensión
TargetLevel: target Level para el algoritmo.
AffCells: cantidad de celdas afectadas
Formula1: valor de la p-excepcion
Formula2: no utilizado
Formula3: valor de la l-excepcion

Formula4: valor de la f-excepcion
TiempoMDX: tiempo que insumió el recálculo de las celdas afectadas
TiempoTot: tiempo total de procesamiento de la revisión, sin tener en cuenta la generación del cubo resultado en el Analysis Server
TiempoFull: no utilizado
Source: cantidad de celdas Source a considerar.

MDTestDet contiene la siguiente información:

TabName: identificador del test
aLevel: nivel considerado
Members: cantidad de miembros del nivel
MemberSel: cantidad de miembros seleccionados en ese nivel.
PathsRevisados: cantidad de paths levantados en ese nivel
FIO: factor que indica el fan in out de un nivel respecto de su nivel inferior.

Apéndice B Estructura de la aplicación

B.1 Componente del servidor

Las unidades que componen el servidor son las siguientes:

ADOMD_TLB (ADOMD_TLB.PAS): es una unit de conversión de la type library ADOMD en Delphi
ADODB_TLB (ADODB_TLB.PAS): es una unit de conversión de la type library ADOMD en Delphi
URevision (URevision.pas): definiciones y funciones referidas a una Revisión, ya sea de un cubo o de una dimensión.
UServerCom (UServerCom.pas): algoritmos y estructuras para la revisión de una dimensión.
TParser (TPARSER.PAS): parser de las cláusulas WHERE de las revisiones.
NString (nstring.pas): rutinas auxiliares para manejo de strings.
UServerCube (UServerCube.pas): algoritmos y estructuras para la revisión de un cubo.
UServerStruc (UServerStruc.pas): definición de estructuras auxiliares para la revisión de Cubos.
URevModel (URevModel.pas): definiciones y funciones para el manejo de modelo de revisiones.
UCubeMgr (UCubeMgr.pas): procesos de creación de cubos resultado en el Analysis Services

B.2 Interfaz de Cliente

Los formularios que componen la interfaz son:

UFrmMain {FPrincipal}: formulario principal.
UFrmRevise {FRevision}: este formulario permite parametrizar la revisión de un cubo o una dimensión.
UFrmResult {FResultados}: este formulario permite seleccionar algún identificador de cubo ya revisado para ver los resultados que se obtuvieron en la última revisión.
UFrmProper {FProperties}: a este formulario se accede dando doble click del mouse en la columna **Cláusula Body** de la Grilla del panel inferior del formulario principal. Muestra los Member Properties del nivel activo, si es que tiene alguna propiedad asociada.
UFrmModelos (FModelos}: este formulario es para seleccionar un Modelo de revisión ya almacenado.
UFrmConfig {FConfig}: este formulario permite la configuración de conexión con el servidor SQL, asume que el Servidor SQL Server tiene los mismos datos de conexión que el Analysis Server
UFrmClock: formulario para mostrar el mensaje de "Sistema procesando...".

UFrmCopyRev: este formulario sirve para ingresar los datos necesarios para copiar revisiones entre distintos cubos.

Las unidades que componen la interfaz son:

ADOMD_TLB (ADOMD_TLB.PAS): es una unit de conversión de la type library ADOMD en Delphi
ADODB_TLB (ADODB_TLB.PAS): es una unit de conversión de la type library ADOMD en Delphi
UMultiSpace (UMultiSpace.pas): definiciones y funciones para el acceso de la interfaz a los objetos de ADOMD.
UFrmUtils (UFrmUtils.pas): funciones auxiliares utilizadas en los Forms.
URevision (URevision.pas): definiciones y funciones referidas a una Revisión, ya sea de un cubo o de una dimensión.
URevModel (URevModel.pas): definiciones y funciones para el manejo de modelo de revisiones.

B.3 Modelo de datos para el desarrollo

Bases de datos

Para almacenar la información de definiciones de las revisiones, y reglas, resultados de la ejecución de revisión de Dimensiones y cubos, así también como generación de Fact Table's para finalmente armar un cubo revisado, se utiliza una base de datos que denominamos OLAPDB.

Por otro lado se envían acompañando esta tesis, bases de datos de ejemplo, como ser una base de datos relacional, Exmple1.Bkp y una multidimensional Example2.Cab las que tendrán que ser restauradas para ejecutar la aplicación (ver apéndice C)

Tablas

MDRevisions: se almacenan los nombres de las revisiones y su relación con los Cubos y Dimensiones.
MDLevels: aquí se almacenan los niveles de la dimensión a revisar y su relación con las columnas de la matriz Cond, utilizada por el algoritmo.
MDRules: aquí se almacenan las reglas de la revisión y su relación con las filas de la matriz Cond, utilizada por el algoritmo.
MDNodes: se almacenan los datos de la intersección entre niveles y reglas, por ejemplo si en esa posición se almacena una cláusula Head o Body.

DistCalcCoord: en esta tabla se almacenan de manera opcional, la información generada por el algoritmo referida a los distcalccoord, quiere decir a las Coordenadas afectadas por la revisión y sus niveles Source.
MDPaths: se almacenan los resultados de los paths revisados. La grabación de esta tabla es opcional.
ResultList: se almacenan los resultados del algoritmo, tiempos, cantidad de los test realizados.
MDModelsHead: Se almacenan datos del cubo, método de revisión, etc., de un modelo de revisión.
MDModelsDet: aquí se almacenan los detalles de los modelos, la relación entre Dimensión – Target level – Revisión.
MDTestHead: se almacena información de la revisión para realizar los análisis y los gráficos.
MDTestDet: se almacena información detallada por nivel, de la revisión para realizar los análisis y los gráficos.

B.4 Objetos definidos en la interfaz de las unidades

A continuación listaremos las interfaces de las unidades utilizadas en el desarrollo de los componentes para la revisión de cubos. Aquí no aparecerán detalles referidos a la interfaz de usuario. Los formularios desarrollados se muestran en el apéndice A. La descripción de cada una de las unidades se vió en el punto B.1.

UMultiSpace

```
TMDObject = class(TObject)
     DS: WideString;
     Catalog1: ICatalog;
     RuleIndex : Integer;
     DimIndex : Integer;
     CubeIndex : Integer;
     aConnData : TConnData;
     constructor create;
     destructor destroy; override;
     procedure CreateCatalog;
     procedure ConnectCatalog(aSrv, aUsr, aPass, aData :string);
     procedure FreeCatalog;
private
public
end;

TDimension = class(TObject)
     ActiveCatalog : TMDObject;
     ActiveRevision : TRevision;
     Name : string;
     procedure NewRevision;
     procedure LevelList(var aCbo : TDataArray; IncludeAll :
                         boolean; ThisDim : integer);
```

```
     procedure PropList(var aCbo : TDataArray; aLevel : string);
     procedure FillRevisions(aCube, aSelDimension : string; var
               aCbo : TDataArray; aWithNew : boolean);
     procedure FillRules(aRevId : integer; var aCbo :
                         TDataArray);
private
public
end;

TCube = class(TObject)
     ActiveCatalog : TMDObject;
     Name : string;
     procedure DimList(var aCbo : TDataArray);
private
public
end;

TMultiSpace = class(TObject)
     ActiveCatalog : TMDObject;
     ActiveCube : TCube;
     ActiveDimension : TDimension;
     constructor create;
     destructor destroy; override;
     procedure CubeList(var aCbo : TDataArray);
private
public
end;
```

URevision

```
TDataArray = array of string;
TBiDataArray = array of TDataArray;
TIntDataArray = array of integer;
TIntBiDataArray = array of TIntDataArray;

TConnData = class(TObject)
     aConn : TADOConnection;
     aMDConn : TADOConnection;
     GlobQuery : TADOQuery;
     aServer : string;
     aUser : string;
     aPassword : string;
     aMDDataSource : string;
     aDataSource : string;
     procedure ADOMDConnect(aSrv, aUsr, aPass, aData : string);
     procedure ADOMDClose;
     procedure OLEDBConnect(aSrv, aUsr, aPass, aData : string);
     procedure OLEDBClose;
     procedure InitGlobalQuery(aConnection : TADOConnection);
     procedure EndGlobalQuery;
private
public
end;
```

```
PTRules = ^TRules;
PTLevels = ^TLevels;

TRules = class(TObject)
     aName : string;
     aCondRow : integer;
private
public
end;

TARules = array of TRules;
TRuleMatrix = array of TARules;

TLevels = class(TObject)
     aName : string;
     aCondCol : integer;
     aVariable : string;
private
public
end;

TRuleNode = class(TObject)
     aType : string;
     oRule : TRules;
     oLevel : TLevels;
     aTerm : string;
     aValue : string;
private
public
end;

TNodeMatrix = array of array of TRuleNode;
PTRevision = ^TRevision;

TRevision = class(TObject)
     ActiveConn : TConnData;
     aName : string;
     aDimension : string;
     aCube : string;
     aRules : array of TRules;
     aLevels : array of TLevels;
     aCond : TNodeMatrix;
     RevId : integer;
     function Save(aUpdate : boolean) : integer;
     function Load : integer;
     function RevisionExist(var aRevId : integer) : boolean;
     function CreateCompleteRule(pName : string; pRow : integer;
                    pLevel : string ; pHeadBody, pHeadValue :
               string; aBiData : TBiDataArray) : PTRules;
     function CreateRule(pName : string; pRow : integer) :
                    integer;
     function CreateLevel(pName, pVar : string; pCol : integer)
               : integer;
```

```
      function CreateRuleNode(pType, pTerm, pValue : string;
                        aRow, aCol : integer) : integer;
      function ExistLevel(pLev : string; var LevNum : integer) :
                  boolean;
      function ExistRule(pName : string; var LevNum : integer) :
                  boolean;
      function ModifLevel(pName, pVar : string; pCol : integer) :
                        integer;
     function ModifRule(pName : string; pRow : integer) :
                  integer;
     function ModifRuleNode(pType, pTerm, pValue : string; aRow,
                  aCol : integer) : integer;
private
public
end;
```

URevModel

```
TRevModel = class(TObject)
      RevName : string;
      ActiveConn : TConnData;
      aCube : string;
      aMeasure : string;
      CatalogMethod : string;
      aArrDim : TDataArray;
      aArrTargLev : TDataArray;
      aArrRev : TDataArray;
      constructor create(aRevName : string; aActiveConn :
            TConnData);
      destructor destroy; override;
      procedure SaveModel;
      procedure LoadModel;
      procedure DeleteModel;
      procedure GetModels(aCube : string; var ArrMod :
                  TDataArray);
      function CheckModel(RevCube : string) : boolean;
private
public
end;
```

UServerStruc

```
TKeyColStruc = record
      LevName : string;
      Levels : TDataArray;
end;

TFromClause = record
      FromClause : string;
      aKeyCol : Array of TKeyColStruc;
end;
```

```
TPathLevels = record
     Value : string;
     OldValues : string;
     aMod : boolean;
     sCond : array of TRuleNode;
end;

TPaths = record
     PathId : WideString;
     PathIdMod : WideString;
     PRulesIndex : integer;
     pLevels : array of TPathLevels;
end;

TAPaths = array of TPaths;

TARevision = array of TRevision;

Tprocedure = procedure(aMsg : string);
```

UServerCom

```
TServerDimension = class(TObject)
     ActiveConn : TConnData;
     aDimension : string;
     aCube : string;
     sRules : TRuleMatrix;
     eRules : TRuleMatrix;
     pRules : TIntBiDataArray;
     rPaths : array of TAPaths;
     TargetPaths : TAPaths;
     Paths : TAPaths;
     aRev : TARevision;
     EFormula : TEFormula;
     Catalog : iCatalog;
     NextPRulesIndex : integer;
     f : text;
     aTabname : string;
     ConsPathsInDim : longint;
     ConsPathsInPaths  : longint;
     ConsPathsLastLevel : longint;
     TotalTime : single;
     SaveTime : single;
     LastLevelCount : longint;
     CatalogMethod  : string;
     VSConn : TConnection;
     VSRs : _RecordSet;
     constructor create(aTempParam, pCube, pDimName, CatMethod,
          pServer, pUser, pPassWord, pDataSource,
          pMDDataSource, aUsr, aPass, aSrv, aData : string;
          pShowResults : integer; GenLog, pSPaths :
     boolean);
     destructor destroy;override;
```

```
      procedure AddRevision(pRev : string);
      function ReviseDimension : integer;
      procedure InitEySRules;
      function NotIn(pName : string; aArr : TARules; ActRule,
                ActLev : integer; CkeckBody : boolean) : boolean;
      procedure ClearVariables;
private
      LastPaths : integer;
      function RevisePaths (plevel : integer) : integer;
      function Merge (plevel, ppath : integer; ppath2 : string) :
                integer;
      function PutNulls (ppath , plevel : integer) : integer;
      function TestConflicts(pRule, pPath : integer) : boolean;
      function ActivatePaths(pLevel : integer) : integer;
      function DeletePaths (plevel : integer) : integer;
      function GetPaths(pRule, pLevel : integer; var aArr :
                TDataArray; var bArr : TIntDataArray) : integer;
      function GetPathsOfDim(pRule, pLevel : integer; var aArr :
           TAPaths) : integer;
      function IsPathInPaths(pPaths : WideString) : boolean;
      function NotInEySRules(pLevel, pRule : integer) : boolean;
      function GetNotrPaths(pRule ,pLevel : integer; var aArr :
           TDataArray; var bArr : TIntDataArray) : integer;
      function IsEmpty(var nRules : TIntDataArray) : boolean;
      function GetHeadCoord(pRule : integer; var pLevel :
           integer; var aCoord : variant) : integer;
      function DeletepRules(pRule, pPath : integer) : integer;
      function DeleterPaths(pRule : integer; pPath : Widestring)
           : integer;
      function DeletePath(pPath : Widestring) : integer;
      function GetTargPaths(aArr : TAPaths; pRule, pLevel :
                integer) : TPaths;
      function InArr(var aArr : TIntDataArray; pValue : integer)
           : integer;
      procedure InitIntArray(var aArr : TIntDataArray; aLen,
                aIniVal : integer);
      procedure  InitIntBiArray(var aArr : TIntBiDataArray; aLen1,
                aLen2, aIniVal : integer);
      function EvalPath(aSentence : string; aLev : integer; aMem
           : Member; pValue : string) : integer;
      function OutputPaths : integer;
      function GetPathsLevels(aLev : Member; var aPosit :
           integer; var aArr : TAPaths; LevelFrom, pRule,
      pLevel : integer) : integer;
      function GetPathsOfDimTP(pLevel : integer; var aArr :
                TDataArray; var bArr : TAPaths) : integer;
      function GetPathInd(pPath : string) : integer;
      procedure printvectors(aMsg : string);
      procedure ListAll(aMem : Member; aLevel : integer; var
                aPosit : integer; var aArr : TAPaths; LevelFrom,
                pRule, pLevel : integer);
      function NotInCoords(aArr : TDataArray; aCd : string) :
                boolean;
      procedure SavePaths;
```

```
     procedure FillLastLevel;
     function InRules(pLevel, pRule : integer; pVect :
               TRuleMatrix) : boolean;
     procedure SaveResults;
     procedure ActMessages(aMsg : string);
public
end;
```

UServerCube

```
TElemDistCoord = record
     Source : string;
     Level : integer;
end;

TArrElemDistCoord = array of TElemDistCoord;

TDistCoords = record
     Value : string;
     LevUniqueName : string;
     aLost : TArrElemDistCoord;
     aGained : TArrElemDistCoord;
end;

TRulesMetrics = record
     aMembers : longint;
     aPaths : longint;
     AllMembers : longint;
end;

TTCoord = record
     Old : string;
     New : string;
     Source : string;
     Level : integer;
end;

TServerCubeInst = class(TObject)
     ActiveConn : TConnData;
     Catalog : iCatalog;
     CatalogMethod  : string;
     aCube : string;
     aDimension : string;
     sRules : TRuleMatrix;
     eRules : TRuleMatrix;
     pRules : TIntBiDataArray;
     rPaths : array of TAPaths;
     TargetPaths : TAPaths;
     Paths : TAPaths;
     Body : TIntDataArray;
     Head : TIntDataArray;
```

```
      aaRules : TIntBiDataArray;
      rCount : TIntBiDataArray;
      lPaths : array of TAPaths;
      DistCalcCoord : array of TDistCoords;
      mCoord : array of TTCoord;
      LevelMin : TIntDataArray;

      aRev : TARevision;
      EFormula : TEFormula;
      NextPRulesIndex : integer;
      ConsPathsInDim : longint;
      ConsPathsInPaths  : longint;
      TotalTime : single;
      VSConn : TConnection;
      VSRs : _RecordSet;
      VSRs2 : _RecordSet;
      TargetLevel : integer;
      aTabname : string;
      ShowResults : integer;
      TimeGetPaths : single;
      TimeGenerateDistCalc : single;
      TimeSavePaths : single;
      TimeSaveDistCalc : single;
      ConsPathsLastLevel : longint;
      LastLevelCount : longint;
      RevStr : string;
      HasError : integer;
      CantDistCalcCoords : LongInt;
      RuleMet : array of TRulesMetrics;
      constructor Create;
      procedure AddRevision(pRev : string);
      procedure InitEySRules;
      procedure InitHeadyBody;
      function NotIn(pName : string; aArr : TARules; ActRule,
                  ActLev : integer; CkeckBody : boolean) : boolean;
private
      LastPaths : integer;
      function RevisePaths (plevel : integer) : integer;
      function Merge (pRule, plevel, ppath : integer; ppath2 :
            string) : integer;
      function PutNulls (ppath , plevel : integer) : integer;
      function TestConflicts(pLevel, pPath : integer) : boolean;
      function ActivatePaths(pLevel : integer) : integer;
      function DeletePaths (plevel : integer) : integer;
      function GetPaths(pRule, pLevel : integer; var aArr :
                  TDataArray; var bArr : TIntDataArray) : integer;
      function GetPathsOfDim(pRule, pLevel : integer; var aArr :
            TAPaths) : integer;
      function IsPathInPaths(pPaths : WideString) : boolean;
      function NotInEySRules(pLevel, pRule : integer) : boolean;
      function GetNotrPaths(pRule ,pLevel : integer; var aArr :
            TDataArray; var bArr : TIntDataArray) : integer;
      function IsEmpty(var nRules : TIntDataArray) : boolean;
```

```
      function GetHeadCoord(pRule : integer; var pLevel :
            integer; var aCoord : variant) : integer;
      function DeletepRules(pRule, pPath : integer) : integer;
      function DeleterPaths(pRule : integer; pPath : Widestring)
            : integer;
      function DeletelPaths(pLevel : integer; pPath : Widestring)
            : integer;
      function DeletePath(pPath : Widestring) : integer;
      function GetTargPaths(aArr : TAPaths; pRule, pLevel :
            integer) : TPaths;
      function InArr(var aArr : TIntDataArray; pValue : integer)
            : integer;
      procedure InitIntArray(var aArr : TIntDataArray; aLen,
            aIniVal : integer);
      procedure InitIntBiArray(var aArr : TIntBiDataArray; aLen1,
            aLen2, aIniVal : integer);
      function EvalPath(aSentence : string; aLev : integer; aMem
            : Member; pValue : string) : integer;
      function GetPathsLevels(aLev : Member; var aPosit :
            integer; var aArr : TAPaths; LevelFrom, pRule,
       pLevel : integer; UniqueName : string) : integer;
      function GetPathsOfDimTP(pLevel : integer; var aArr :
            TDataArray; var bArr : TAPaths) : integer;
      function GetPathInd(pPath : string) : integer;
      procedure ListAll(aMem : Member; aLevel : integer; var
            aPosit : integer; var aArr : TAPaths; LevelFrom,
            pRule, pLevel : integer);
      function NotInCoords(aArr : TDataArray; aCd : string) :
            boolean;
      procedure SaveDistCalcCoords;
      procedure FillLastLevel;
      function InRules(pLevel, pRule : integer; pVect :
            TRuleMatrix) : boolean;
      procedure CheckmCoord;
      procedure CheckDistCalcCoord;
      procedure AddCoord(pPath : integer; OldC, NewC, aSource :
            string; plevel : integer; var t : TTCoord);
      function InDistCalcCoord(aValue : string) : integer;
      procedure AddTupleToLost(t : TTCoord; pLevUniqueName :
            string);
      procedure AddTupleToGained(t : TTCoord; pLevUniqueName :
            string);
      procedure ActMessages(aMsg : string);
      procedure ClearVariables;
      procedure SavePaths;
      procedure SaveResults;
      procedure SaveTestHead;
public
end;

TServerCube = class(TObject)
      aSrvCubeInst : TServerCubeInst;
      ActiveConn : TConnData;
      aCube : string;
```

```
Catalog : iCatalog;
f : text;
aTabname : string;
TotalTime : single;
CatalogMethod  : string;
aMeasure : String;
aArrDim : TDataArray;
aArrTargLev : TDataArray;
aArrRev : TDataArray;
AffCellRS : array of _RecordSet;
AffCellModL : array of array of double;
AffCellModG : array of array of double;
VSConn : TConnection;
DistCalcCoords : array of array of TDistCoords;
aFromClause : TFromClause;
ShowResults : integer;
CountAffectedCells : longint;
TimeGetAffectedCells : single;
TimeUpdateAffectedCells : single;
TimeCreateCube : single;
TimeProcessCube : single;
CalculateMDXTime : single;
CantidadMDX : longint;
Entro : boolean;
FirstRS : integer;
PreDataSource : string;
AllCells : longint;
HasError : integer;
GenerateLog : boolean;
GenerateCube : boolean;
pSavePaths : boolean;
pGenFactTable : boolean;
CountSourceCells : longint;
aArrGetAff : TIntDataArray;
aArrGetSource : TIntDataArray;
aOnTest : boolean;
constructor create(aTempParam, pServer, pUser, pPassWord,
      pDataSource, pMDDataSource, aUsr, aPass, aSrv,
 aData : string; pShowResults : integer; GenLog,
 GenCube, pSPaths, GenFT, OnTest : boolean);
destructor destroy;override;
function ReviseCube : integer;
procedure GetAffectedCells;
procedure UpdateAffectedCells;
procedure DeleteDistCalcCoordsAndPaths;
procedure OpenMDConnection;
procedure CopyDistCalcCoords(aInd : integer);
procedure GetCellNewValue(aIndex : integer; LostOrGained :
      integer; var pLost, pGained : double);
function GetDCCIndex(pCoord : string; aInd : integer) :
           integer;
procedure CreateFactTable(pTabName : string; pFields :
           TDataArray);
procedure ClearCubeVariables;
```

```
     procedure SaveResults;
     function  GetPosFields(pName  :  string;  pFields  :  TDataArray)
                : integer;
     function GetArrDimIndex(aDim : string) : integer;
     procedure DeleteResults;
private
     procedure printvectors(aMsg : string);
     procedure PrintFirstVectors;
     procedure PrintLastVectors;
public
end;
```

UCubeMgr

```
TCubeMgr = class(TObject)
     ActiveConn : TConnData;
     aCube : string;
     aDataSource : string;
     aDataBase : string;
     aMeasure : string;
     aArrDim : TDataArray;
     aArrTargLev : TDataArray;
     aFromClause : TFromClause;
     StrConn : widestring;
     mSourceTable : string;
     mSourceColumn : string;
     mFromClause : string;
     mJoinClause : string;
     TimeProcess : single;
     aDatabases : TDataArray;
     HasError : integer;
     constructor create(pCube, pMeasure : string; aActiveConn :
          TConnData);
     function CreateCube : integer;
     function UpdateCube : integer;
     function GetDatabases : integer;
private
public
end;
```

TParser

```
TLexem = class(TObject)
protected
     constructor Create;
end;

TNumLexem = class(TLexem)
private
     fValue : double;
public
```

```
      constructor Create(Value : double);
      property Value : double read fValue write fValue;
end;

TStrLexem = class(TLexem)
private
      fValue : string;
public
      constructor create(const Value : string);
      property Value : string read fValue write fValue;
end;

TLogLexem = class(TLexem)
private
      fValue : boolean;
public
      constructor create(const Value : boolean);
      property Value : boolean read fValue write fValue;
end;

EFormulaError = class(Exception);

TEFormula = class(TObject)
      Eq0           : PChar;
      Eq            : PChar;
      LastEq        : PChar;
      SValue        : string;
      NValue        : double;
      LValue        : boolean;
      GetData       : Boolean;
      Tipo          : string[1];   {'S', 'N', 'L'}
      Catalog       : iCatalog;
      ActEvalMember : Member;
      ActEvalLevel : integer;
      ActValue : string;
      aDimension : string;
      aat100 : single;
      constructor create(EqE : PChar; aCateg : string);
      destructor destroy; override;
      procedure UnGet;
      procedure RaiseError(const Msg : string);
      function GetChar : char;
      function Pot(n1, n2 : double) : double;
      function GetToken (var lex : TLexem) : integer;
      function Unity : TLexem;
      function Factorial : TLexem;
      function Potencia : TLexem;
      function Termino : TLexem;
      function Factor : TLexem;
      function FactorLog : TLexem;
      function Expresion : TLexem;
      function ExpresionLog : TLexem;
      function EvaluaFormula(aGetData : boolean) : boolean;
      procedure EvalFunc(aParam : variant; var aRet : string);
```

```
private
public
      aPosError : Integer;
      Archivo   : string;
end;

TPStr  = ^string;
```

Apéndice C Instrucciones de instalación de la aplicación

Versión del componente a instalar

La versión actual de la aplicación esta desarrollado como herramienta de testing en un componente único, en el futuro la intención sería separar la interfaz de usuario y el componente, y que la parte que va al servidor corra como un componente COM+. La aplicación esta preparada como para hacer esta separación, sólo hay que hacerla y testear la instalación y las interfaces.

Software del Servidor

En la versión actual, la aplicación deberá instalarse sobre el servidor de OLAP que deberá tener el siguiente software instalado:

Microsoft Windows 2000 server Spanish
Windows 2000 Server Service Pack 2
Microsoft SQL Server 2000 Ingles Enterprise Edition
SQL Server 2000 Service Pack 2
Microsoft SQL Server 2000 Ingles Analysis Services
SQL Server 2000 Analysis Services Service Pack 2

Actualmente se recomienda ejecutarlo con una resolución de pantalla de mínimo 800 x 600

Bases de datos a instalar en el servidor.

Utilizando el Enterprise Manager de SQL Server 2000, bajar el backup que se encuentra en el archivo Example1Relac.bkp y el backup OlapDB.bkp en el subdirectorio \BASES.

Utilizando el Analysis Manager del Analysis Server 2000, con la opción Restore Database, bajar el backup que se encuentra en el archivo Example2MD.cab en el subdirectorio \BASES.

Los DataSources de los cubos del backup que se envían, están apuntando al servidor donde fueron creados, por lo que con el Analysis Manager habría que cambiarlos al servidor donde se instaló la aplicación.

Si no funciona, habría que copiar el restore.bat a C:\Archivos de programa\Microsoft Analysis Services\Bin y luego ejecutarlo parado en ese directorio, desde la línea de comandos, con el parámetro del .CAB con la ruta completa.

Ejecutar el SetUp de la aplicación desde el directorio raíz del CD, elegir como directorio destino, un directorio del servidor.

Referencias:

[BDel5] *Imprise Corporation, Borland Delphi Help, 1999.*

[Bru99] Andrew J. Brust. *Put OLAP and ADO MD to Work*, www.vbpj.com August 1999.

[Fir98] Joseph Firestone. *Architectural Evolution in Data Warehousing and Distributed Knowledge Management Architecture.* Paper No. Eleven, *1998* .

[HMV99a] C. Hurtado, A. Mendelzon, and A. Vaisman. *Maintaining Data Cubes under Dimension Updates*. In Proceedings of the IEEE-ICDE Conference, pages 346-355, March 1999.

[HMV99b] C. Hurtado, A. Mendelzon, and A. Vaisman. *Updating OLAP dimensions*. In Proceedings of the 2nd IEEE-DOLAP Workshop, Kansas City, Missouri, USA, November 1999.

[HTTPDJ] Database Journal. *Introduction to SQL Server 2000 Analysis Services*, http://www.databasejournal.com/features/mssql/article.php/1429671.

[HTTPMDX] Microsoft Coporation. *Manipulate and Query OLAP Data Using ADOMD and Multidimensional Expressions*, http://www.microsoft.com/msj/0899/mdx/mdx.htm, August 1999

[Inm92] William H. Inmon. *Building the Data Warehouse.* Wiley-QED Jhn Wiley & Sons Inc., 1992.

[Mat02] Malcolm Matthews. *Writing MTS and COM+ Components in Delphi* http://consulting.dthomas.co.uk, 2002.

[MSHLP99] Microsoft Corporation. *Microsoft Help, OLE DB for OLAP*, July 1999.

[MSSQL2k] Microsoft Corporation, *SQL Server Books Online*, 2000.

[MSSQLAS] Microsoft Corporation, *SQL Server Analysis Services Books Online*, 2000.

[MV01] M. Minuto Espil and A. Vaisman. *Efficient intentional redefinition of aggregation hierarchies in Multidimensional Databases*, ACM-DOLAP'01, 2001.

[MV02] M. Minuto Espil and A. Vaisman. *Revising Data Cubes with Exceptions: A Rule-Based Perspective,* DMDW 2002: 72-81.

[MV03] M. Minuto Espil and A. Vaisman. *Revising aggregation hierarchies in OLAP: a rule-based approach,* Data & Knowledge Engineering vol 45/2 pág. 225 – 256 Advances in

online analytical processing, Edited by J. Hammer, Elsevier Science Journals, North Holland, Amsterdam, Holanda, Mayo 2003

[Vas00] P. Vassiliadis. *Gulliver in the land of data warehousing: practical experiences and observations of a researcher*. In Proceedings of DMDW'2000, 2000.

[WB97] M.C. Wu, A.P. Buchmann. R*esearch Issues in Data Warehousing*. BTW'97, 1997.

[Wid95] J. Widom. *Research Problems in Data Warehousing*. In Proc. 4th Int. Conf. on Information and Knowledge Management, Nov. 1995.

[You99] Sakhr Youness. *Using MDX and ADOMD to Access Microsoft OLAP Data*, Commerce One 1999.

[ZR99] X. Zhang and E. A. Rundensteiner. *Data Warehouse Maintenance Under Concurrent Schema and Data Updates*. In Proceedings of IEEE International Conference on Data Engineering, 1999.