

***STARCUBE: Una herramienta ROLAP de análisis multidimensional para el soporte a la toma de decisiones, débilmente acoplada con el SGBD PostgreSQL.***

**Álvaro Ricardo Cújar Rosero  
Alexander Arturo Mera Caraballo  
Camilo Alberto Villota Cerón**

**Universidad de Nariño  
Facultad de Ingeniería  
Programa de Ingeniería de Sistemas  
San Juan de Pasto  
2009**

***STARCUBE: Una herramienta ROLAP de análisis multidimensional para el soporte a la toma de decisiones, débilmente acoplada con el SGBD PostgreSQL.***

**Álvaro Ricardo Cújar Rosero  
Alexander Arturo Mera Caraballo  
Camilo Alberto Villota Cerón**

**Trabajo de Grado presentado como requisito parcial para optar el título de  
Ingeniero de Sistemas**

**Director del Proyecto  
Ricardo Timarán Pereira Ph.D.**

**Universidad de Nariño  
Facultad de Ingeniería  
Programa de Ingeniería de Sistemas  
San Juan de Pasto  
2009**

"Las ideas y las conclusiones aportadas en el presente trabajo son  
responsabilidad exclusiva de sus autores"

Artículo 1, acuerdo No. 324 de octubre 11 de 1966, emanado por el Honorable  
Consejo Directivo de la Universidad de Nariño.

Nota de aceptación:

---

---

---

---

---

---

---

Firma del presidente del jurado

---

Firma del jurado

---

Firma del jurado

**San Juan de Pasto 24 de Marzo de 2009**

## AGRADECIMIENTOS

A LA VICERRECTORÍA DE INVESTIGACIONES Y POSTGRADOS VIPRI POR  
EL APOYO FINANCIERO DE ESTE PROYECTO.

## RESUMEN

EN ESTE PROYECTO DE INVESTIGACIÓN SE PRESENTA EL ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE **“STARCUBE, UNA HERRAMIENTA ROLAP DE ANÁLISIS MULTIDIMENSIONAL PARA EL SOPORTE A LA TOMA DE DECISIONES, DÉBILMENTE ACOPLADA CON EL SGBD POSTGRESQL”**, ASÍ COMO LAS PRUEBAS DE FUNCIONALIDAD CON REPOSITARIOS DE DATOS REALES.

LA ARQUITECTURA DE STARCUBE ESTÁ COMPUESTA POR TRES MÓDULOS: EL MÓDULO DE INTERFAZ GRÁFICA DE USUARIO (GUI), EL MÓDULO DE KERNEL Y EL MÓDULO DE UTILIDADES.

EL MÓDULO DE GUI PERMITE VISUALIZAR Y MANIPULAR DE MANERA AMIGABLE LA HERRAMIENTA STARCUBE, EL MÓDULO DE KERNEL ES EL NÚCLEO DE STARCUBE DONDE SE HACEN LA DEFINICIÓN Y MANIPULACIÓN DE CUBOS Y FINALMENTE EL MÓDULO DE UTILIDADES QUE PERMITE LA CONEXIÓN CON EL SGBD POSTGRESQL Y LAS BASES DE DATOS.

## ABSTRACT

IN THIS RESEARCH PROJECT IS PRESENTED THE ANALYSIS, DESIGN AND IMPLEMENTATION OF **“STARCUBE, A ROLAP TOOL FOR MULTIDIMENSIONAL ANALYSIS TO SUPPORT DECISION MAKING, WEAKLY COUPLED WITH THE POSTGRESQL DBMS”** AND THE FUNCTIONALITY TESTS WITH REAL DATA REPOSITORIES.

STARCUBE ARCHITECTURE CONSISTS OF THREE MODULES: THE GRAPHICAL USER INTERFACE MODULE (GUI), THE KERNEL MODULE AND UTILITIES MODULE.

THE GUI MODULE ALLOWS TO VIEW AND TO MANIPULATE THE STARCUBE TOOL IN A FRIENDLY WAY, THE KERNEL MODULE IS THE CORE OF STARCUBE WHERE THE DEFINITION AND MANIPULATION OF THE CUBES ARE DONE AND FINALLY, THE UTILITY MODULE THAT ALLOWS THE CONNECTION TO THE POSTGRESQL DBMS AND THE DATABASES.

## ÍNDICE GENERAL

INTRODUCCIÓN	14
1. BODEGAS DE DATOS Y SISTEMAS DE PROCESAMIENTO ANALÍTICO EN LÍNEA	18
1.1 DATA WAREHOUSE	18
1.1.1 Características de un data warehouse	18
1.1.2 Arquitectura de un data warehouse	18
1.1.3 Data warehousing	19
1.2 TECNOLOGÍA OLAP	20
1.2.1 Arquitecturas de integración de las herramientas OLAP con un SGBD.	20
1.3 SISTEMAS ROLAP, MOLAP, HOLAP	21
1.3.1 ROLAP	21
1.3.2 MOLAP	22
1.3.3 HOLAP.	22
1.4 MODELO MULTIDIMENSIONAL	23
1.4.1 Estructura.	23
2. ANÁLISIS DE HERRAMIENTAS OLAP	28
2.1 MONDRIAN PENTAHOTM	28
2.1.1 Creador de cubos, PentahoTM Cube Designer v.0.7.2	38
2.2 OLAP BROWSER	44
2.3 OLAPX v.4	48
2.3.1 Creador de cubos OLAPx Acc	55
3. METODOLOGIA, LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE STARCUBE	66
3.1 METODOLOGÍA DE DESARROLLO	66
3.2 METODOLOGÍA DE ANÁLISIS Y DISEÑO	67
3.3 LENGUAJE UNIFICADO DE MODELADO	68
3.4 LENGUAJE DE PROGRAMACIÓN	73
3.4.1 Principales características de JavaTM.	73
3.5 ENTORNO DE DESARROLLO NETBEANS IDE 6.0	75
3.6 CONTROLADOR JDBC	75
3.7 JFREECHART	77
3.8 OLAP4J	77
3.9 MYDOGGY	83
3.10 BIBLIOTECA GRÁFICA SWING DE JAVA™	85
4. ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE LA HERRAMIENTA STARCUBE	88
4.1 ANÁLISIS DE STARCUBE	88
4.1.1 Funciones	89
4.1.2 Casos de uso	90
4.1.3 Diagramas de casos de uso	93
4.2 DISEÑO DE STARCUBE	96



4.2.1 Diagramas de secuencia	96
4.2.2 Diagramas de paquetes	103
4.2.3 Diagramas de clases STARCUBE	106
5. IMPLEMENTACIÓN	130
5.1 ARQUITECTURA	130
5.2 ESTRUCTURA DE PAQUETES DE LA APLICACIÓN	132
5.2.1 Módulo de kernel	133
5.2.2 Módulo GUI	137
6. PRUEBAS Y RESULTADOS	159
6.1 ANÁLISIS DE FUNCIONALIDAD	159
6.2 DEFINICIÓN DE REQUISITOS PARA LA CONSTRUCCIÓN DEL DATAMART UDENAR	159
6.3 CREACIÓN DE CUBOS	166
6.3.1 Creación de cubo aprobación	166
6.3.2 Creación de cubo población	171
6.3.3 Creación de cubo especial	172
6.4 MANIPULACIÓN DE CUBOS	173
6.4.1 Manipulación de cubo aprobación	173
6.4.2 Manipulación de cubo población	177
6.4.3 Manipulación del cubo especial	178
7. CONCLUSIONES	179
8. RECOMENDACIONES	180
REFERENCIAS	181

## ÍNDICE DE FIGURAS

Figura 2.1. Arquitectura de un data warehouse	19
Figura 2.2. Arquitectura ROLAP débilmente acoplada	21
Figura 2.3. Arquitectura MOLAP	22
Figura 2.4. Muestra de la estructura multidimensional	24
Figura 2.5. Aplicación operaciones roll-up y drill-down sobre un cubo de datos.	25
Figura 2.6. Muestra de la aplicación de las operación slice.	26
Figura 2.7. Muestra de la aplicación de las operación pivot.	27
Figura 2.8. Arquitectura manejador de protocolo nativo tipo 4.	76
Figura 2.9. Arquitectura de la API OLAP4J	81
Figura 2.10. Componentes de la API OLAP4J	82
Figura 2.11. STARCUBE	94
Figura 2.12. Manejo de usuarios	94
Figura 2.13. Creación de cubos	95
Figura 2.14. Manipulación de cubos	95
Figura 2.15. Operaciones	96
Figura 2.16. Add measure table	97
Figura 2.17. Delete measure table	98
Figura 2.18. Delete member	99
Figura 2.19. Operación pivot	100
Figura 2.20. Drill-down, roll-up	101
Figura 2.21. Configuración y creación del esquema del cubo	102
Figura 2.22. Paquete principal	103
Figura 2.23. Paquete core	104
Figura 2.24. Paquete interfaz gráfica	105
Figura 2.25. Cubeconfiguration	106
Figura 2.26. Hierarchy	107
Figura 2.27. Main	108
Figura 2.28. Managercube	109
Figura 2.29. Member	110
Figura 2.30. Operations	111
Figura 2.31. Querymdx	113
Figura 2.32. Tree	114
Figura 2.33. Account	114
Figura 2.34. Cube	116
Figura 2.35. Cubeconfiguration	116
Figura 2.36. Dbconexion	119
Figura 2.37. Dice	120
Figura 2.38. Graphic	121
Figura 2.39. Main	122
Figura 2.40. Measures	124
Figura 2.41. Navigator	125
Figura 2.42. Reports	127
Figura 2.43. Table	128

Figura 2.44. Toolbar	129
Figura 2.45. Arquitectura general de STARCUBE	131
Figura 2.46. Arquitectura interna de STARCUBE	132
Figura 2.47. Autenticación de usuarios	138
Figura 2.48. Marco de trabajo	139
Figura 2.49. Apariencia marco de trabajo	140
Figura 2.50. Usuarios del sistema.	141
Figura 2.51. Edición de usuario	142
Figura 2.52. Creación de usuario.	142
Figura 2.53. Cubos creados.	143
Figura 2.54. Cubos compartidos.	143
Figura 2.55. Nombre del cubo.	145
Figura 2.56. Configuración fuente de datos	146
Figura 2.57. Conexiones y selección de campos.	147
Figura 2.58. Medidas del cubo.	148
Figura 2.59. Dimensiones del cubo.	149
Figura 2.60. Revisión del cubo.	150
Figura 2.61. Permisos del cubo.	151
Figura 2.62. Guardar el cubo.	152
Figura 2.63. Gráfico de barras apiladas	153
Figura 2.64. Gráfico de barras	153
Figura 2.65. Ayuda de usuario.	154
Figura 2.66. Medidas del cubo.	155
Figura 2.67. Navegador del cubo.	156
Figura 2.68. Reporte de un análisis	157
Figura 2.69. Datos del análisis.	157
Figura 2.70. Restricciones cubo.	158
Figura 2.71. Esquema en copo de nieve, cantidad de eventos por región y carrera.	163
Figura 2.72. Esquema en copo de nieve, nota final y número de veces materia cursada por región y carrera.	164
Figura 2.73. Esquema en copo de nieve, cantidad de alumnos por región y carrera	165
Figura 2.74. Configuración conexión datamart.	166
Figura 2.75. Nombre del cubo.	167
Figura 2.76. Conexiones y selección de campos.	168
Figura 2.77. Definición de medidas.	169
Figura 2.78. Definición de dimensiones.	169
Figura 2.79. Revisión del cubo.	170
Figura 2.80. Estructura final cubo aprobación.	171
Figura 2.81. Estructura final cubo población.	172
Figura 2.82. Estructura final cubo especial.	173
Figura 2.83. Comenzar análisis.	174
Figura 2.84. Eliminar medida.	175
Figura 2.85. Drill down.	176

Figura 2.86. Resultado final análisis.	177
Figura 2.87. Resultado final análisis.	178

## **ÍNDICE DE TABLAS**

Tabla 2.1. Funciones de STARCUBE	89
Tabla 2.2. Caso de uso inicio	90
Tabla 2.3. Caso de uso manejo de usuarios	91
Tabla 2.4. Caso de uso creación de cubos	92
Tabla 2.5. Caso de uso manipulación de cubos	92
Tabla 2.6. Caso de uso operaciones	93
Tabla 2.7. Descripción de las tablas de dimensión del datamart UDENAR.	160
Tabla 2.8. Descripción de las tablas de hechos del datamart UDENAR.	161

## **ÍNDICE DE ANEXOS**

ANEXO 1	183
ANEXO 2	184

## GLOSARIO

**DATA WAREHOUSE (BODEGA DE DATOS):** Colección de datos orientados al tema, integrados, temporales y no volátiles para la toma de decisiones.

**DATAMART:** Almacén de datos. Data Warehouse a menor escala.

**DICE (RESTRINGIR):** Operación OLAP de restricción de los datos del cubo.

**DIMENSION:** Categorías descriptivas de un cubo de datos.

**DRILL DOWN (TALADRAR):** Operación OLAP para aumentar el nivel del detalle de los datos del cubo.

**HIERARCHY (JERARQUÍA):** Es como se organizan las dimensiones. Una dimensión puede contener distintas jerarquías.

**HOLAP:** Hybrid OLAP. Clasificación de OLAP en donde se hace uso de SGBD relacionales y SGBD multidimensionales.

**LEVEL (NIVEL):** Es cada una de las sub ramas de las jerarquías.

**MEASURES (MEDIDAS):** Valores cuantitativos de un cubo, a los que se pueden operar aritméticamente.

**MEMBER (MIEMBRO):** Corresponde al elemento de una jerarquía que proporciona una característica a cada uno de los datos del cubo.

**MOLAP:** Variante de OLAP en donde los datos se los toma de un SGBD multidimensional.

**OLAP:** Acrónimo de Procesamiento Analítico en Línea (On-line Analytical Proccesing).

**PIVOT (PIVOTE):** Operación OLAP para cambiar la vista del cubo a otra perspectiva.

**ROLAP:** Relational OLAP. Clasificación de OLAP para SGBD relacionales.

**ROLL UP (ENROLLAR):** Operación OLAP para disminuir el nivel del detalle de los datos del cubo.

**SLICE (REBANAR):** Operación OLAP para agregar o quitar una dimensión en el análisis del cubo.

## INTRODUCCIÓN

El proceso de brindar soporte a la toma de decisiones a nivel estratégico ha sido reconocido por muchos investigadores, como un tema clave de investigación, y por muchas compañías empresariales como una importante área dentro de la inteligencia de negocios [1].

La Inteligencia de Negocios (BI) representa las herramientas y sistemas que juegan un papel clave en el proceso estratégico de la planificación de una compañía. Estos sistemas permiten reunir, almacenar, y analizar los datos corporativos siendo una importante ayuda en la toma de decisiones [20].

Uno de los tipos de sistemas que se utiliza para Inteligencia de Negocios son las herramientas OLAP, con las cuales se realiza un análisis en línea sobre una bodega de datos. OLAP le permite al usuario (analistas, ejecutivos, directivos) acceder a los datos de forma rápida, consistente e interactiva a través de diferentes puntos de vista. Esta tarea no implica procesar los datos por que se desea presentar la dimensionalidad real de la empresa al usuario, con OLAP un usuario podría analizar por ejemplo resultados de ventas e ir profundizando en detalle en cada región, canal de venta, equipos de ventas o vendedores, artículos, hasta encontrar la información importante para la toma de decisiones. Este es un sencillo ejemplo de cómo OLAP ayuda a hacer un análisis correcto y rápido de la información desde diferentes puntos de vista o "dimensiones" según lo que el usuario desee [1] [14] [2].

Las investigaciones en OLAP, se centraron inicialmente en crear un nuevo modelo de datos que soportara estos sistemas: el modelo multidimensional. Investigaciones posteriores se han focalizado en desarrollar un modelo multidimensional para manejar la imprecisión y la incertidumbre, ofreciendo como resultado el desarrollo de herramientas que permiten modelar información de forma imperfecta y ocultar la complejidad subyacente de cara al usuario final del sistema. Una herramienta OLAP debe integrar una variedad de interfaces: interfaces de consulta directa, herramientas para generar reportes, herramientas de análisis exploratorio (gráficos, estadísticas descriptivas, etc.) que juntos ayudan al usuario a tener una mayor producción de ideas en base a los datos almacenados en la bodega de datos [1].

En este documento se presenta el trabajo de grado para optar el título de Ingeniero de Sistemas. El resultado final será: *“STARCUBE: Una herramienta ROLAP de análisis multidimensional para el soporte a la toma de decisiones, débilmente acoplada con el SGBD PostgreSQL.”*, basada en Software Libre y orientada a las pequeñas y medianas empresas.

## TÍTULO

*STARCUBE: Una herramienta ROLAP de análisis multidimensional para el soporte a la toma de decisiones, débilmente acoplada con el SGBD PostgreSQL.*

## LÍNEA DE INVESTIGACIÓN

El presente trabajo de grado, se encuentra inscrito bajo la línea de software y manejo de información. Se enmarca dentro del área de las Bases de Datos.

## ALCANCE Y DELIMITACIÓN

STARCUBE es una herramienta que contempla el desarrollo de una interfaz gráfica amigable para el usuario que permite: Acceso a bodegas de datos, bases de datos, seleccionar dimensiones, hechos y medidas, creación de cubos de datos y análisis de los mismos.

Las pruebas de funcionalidad se realizaron utilizando el SGBD PostgreSQL, con una colección de datos reales, es decir usando la base de datos académica de la Universidad de Nariño.

## PROBLEMA OBJETO DE ESTUDIO

Descripción del problema. Muchos investigadores y empresas han observado la necesidad de representar la información almacenada en bases de datos de una manera fácil de interpretar, para la toma de decisiones de nivel estratégico, haciendo de esta un área activa de investigación [20].

Algunas herramientas como OLAPX [16], OLAPBrowser [12], instantOLAP [13], necesitan de la adquisición de costosas licencias para su utilización. Este hecho limita a las pequeñas y medianas empresas u organizaciones, el acceso de herramientas OLAP para la toma de decisiones, que inciden directamente en la obtención de mayores ganancias y en el aumento de su competitividad.

Por otra parte en el mundo del software libre existe la herramienta: Mondrian Project (versión beta), que a pesar de ser una herramienta orientada a la web muy completa, posee ciertas debilidades tales como su difícil instalación y configuración para el usuario administrador.

Formulación del problema. ¿Cómo facilitar a las pequeñas y medianas empresas la toma de decisiones donde se explote la potencialidad de visualización gráfica?

## OBJETIVOS

Objetivo General. Brindar una alternativa para facilitar la toma de decisiones de las pequeñas y medianas empresas (PYMES) tanto de Colombia como de cualquier parte del mundo, mediante el desarrollo de una herramienta ROLAP de análisis multidimensional débilmente acoplada con el sistema gestor de bases de datos PostgreSQL, bajo lineamientos de Software Libre.

### Objetivos específicos

Estudiar y analizar diferentes herramientas ROLAP existentes en el mercado actual para determinar los requerimientos de la herramienta STARCUBE.

Analizar, diseñar, desarrollar e integrar en una sola herramienta ROLAP un módulo de modelamiento multidimensional con el fin de manipular de manera adecuada los datos; un módulo de visualización y manipulación de los cubos de datos de manera gráfica, con el propósito de mostrarle al usuario la información de manera intuitiva, fácil de asimilar y de manejar; y un módulo de generación de reportes de la información contenida en los cubos de datos, con el objeto de proporcionar portabilidad a la información.

Acoplar débilmente la herramienta ROLAP con el SGBD PostgreSQL con el propósito de extraer los datos con los que trabajará la aplicación STARCUBE.

Realizar las pruebas utilizando la herramienta STARCUBE con la base de datos del sistema académico de la Universidad de Nariño y analizar los resultados para garantizar la confiabilidad de la información generada por la aplicación.



## JUSTIFICACIÓN

Las herramientas ROLAP existentes actualmente en el mercado para el soporte de toma de decisiones son en su gran mayoría software propietario, es decir hay que pagar por su adquisición grandes sumas de dinero lo cual excluiría a las pequeñas y medianas empresas para el uso de este tipo de tecnologías.

Las escasas herramientas ROLAP basadas en software libre existentes actualmente en el mercado se encuentran en versiones Beta y son muy complicadas para los usuarios de instalar y de configurar.

El desarrollo de STARCUBE, como una herramienta ROLAP bajo licencia pública CPL (Common Public License), permite que las pequeñas y medianas empresas, utilicen esta tecnología para mejorar la toma de decisiones, maximicen sus ganancias con decisiones acertadas y eleven su poder competitivo.

La posibilidad de generar nuevas investigaciones a partir de los resultados del presente proyecto.

## ORGANIZACIÓN DEL DOCUMENTO

Este documento se encuentra organizado de la siguiente manera: en el capítulo 1 se especifican los objetivos generales y específicos del proyecto, en el capítulo 2 se presenta los sistemas de procesamiento analítico en línea, en el capítulo 3 se muestra el análisis de de herramientas OLAP, en el capítulo 4 se presentan los lenguajes y herramientas para el desarrollo de STARCUBE, en el capítulo 5 se muestra el análisis, diseño y modelado, en el capítulo 6 se presenta análisis, diseño y construcción de la herramienta STARCUBE, en el capítulo 7 se muestra la implementación de la herramienta, en el capítulo 8 las pruebas y resultados, en el capítulo 9 las conclusiones y en el capítulo 10 las recomendaciones.

## 1. BODEGAS DE DATOS Y SISTEMAS DE PROCESAMIENTO ANALÍTICO EN LÍNEA

Un correcto análisis de los datos permite lograr una toma de decisiones acertada y generar mejores beneficios para las empresas, por esta razón se hace necesaria la utilización de herramientas que ayuden a llevar a cabo este tipo de actividades.

En este capítulo se realiza una breve introducción a los conceptos de DATA WAREHOUSE Y OLAP.

### 1.1 DATA WAREHOUSE

En [4] se identifica los principales problemas surgidos cuando los usuarios quisieron hacer análisis de los datos no estructurados extraídos de las bases de datos transaccionales. Encontrar los datos pertinentes en una consulta era muy difícil y muchas veces había inconsistencias en los datos extraídos.

Así, Inmon propuso el término “Data Warehouse” y sugirió que la finalidad de éstos es hacer datos precisos, consistentes y que sean accesibles por los usuarios de forma eficiente [1]. Una definición de Data Warehouse sería la siguiente:

*“Un **Data Warehouse** es una colección de datos orientados al tema, integrados, temporales y no volátiles para la toma de decisiones” [3].*

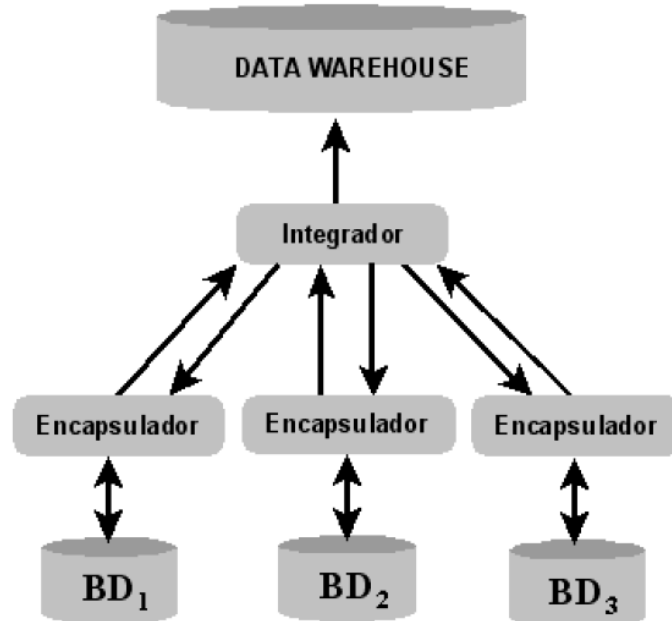
#### 1.1.1 Características de un data warehouse

- Es orientada al tema.
- Esta organizada en torno a los datos más importantes de la empresa.
- Mediante filtros elimina información poco importante.
- La información se la obtiene desde distintas fuentes.
- La información obtenida y almacenada es consistente [7].
- Maneja elevados volúmenes de información.

#### 1.1.2 Arquitectura de un data warehouse

Un sistema de Data Warehouse consta de componentes esenciales, como lo son las fuentes de información, encapsuladores e integradores [1]. En la figura 2.1 se presenta la arquitectura de un Data Warehouse.

Figura 2.1. Arquitectura de un data warehouse



- Entre las fuentes de información se encuentran bases de datos transaccionales y bases de datos no convencionales (bases de conocimiento, bases de datos documentales, HTML, XML, etc).
- El sistema encapsulador se compone de dos módulos: el primero se encarga de trasladar los datos hacia el repositorio y el segundo se encarga de monitorear la fuente de información y de informar si se producen cambios en las fuentes de conocimientos.
- El integrador se encarga de filtrar y unificar la información proveniente de los encapsuladores, de forma que se encuentre disponible en el Data Warehouse. [1].

### 1.1.3 Data warehousing

Las compañías al intentar implementar un modelo de Data Warehouse se dieron cuenta que no se trataba de una pieza de software que se pudiera adquirir, sino más bien un proceso de reingeniería de la información que se forma con la organización [1], es por esta razón que algunos autores introdujeron el término de Data Warehousing y lo definen como un proceso para construir Data Warehouse, además de que implica la existencia de funciones que den soporte a la creación y

mantenimiento de acceso a los usuarios finales para obtener datos completos y consistentes de la empresa con el propósito de responder preguntas de negocios y tomar decisiones, de una forma que no era posible hasta ahora. Es en este punto donde surge uno de los principales enfoques tecnológicos orientado a la toma de decisiones empresariales: **OLAP** (*On-line Analytical Proccesing*).

## 1.2 TECNOLOGÍA OLAP

Definición de OLAP. *“OLAP es una categoría de tecnología software que permite a los analistas, directivos y ejecutivos acceder a los datos de forma rápida, consistente e interactiva a través de una amplia variedad de vistas de la información que han sido obtenidas de datos sin procesar para reflejar la dimensionalidad real de la empresa como la entiende el usuario”* [14].

### 1.2.1 Arquitecturas de integración de las herramientas OLAP con un SGBD.

Las arquitecturas de integración de las herramientas OLAP con un SGBD se pueden ubicar en una de tres tipos: herramientas débilmente acoplados, medianamente acoplados y fuertemente acoplados con un SGBD [19].

Una arquitectura es débilmente acoplada cuando los algoritmos de análisis en línea y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de una interfaz [19].

Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de análisis en línea se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario [19].

Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de análisis en línea forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de análisis en línea y posibilitándolo para desarrollar aplicaciones de este tipo [19].

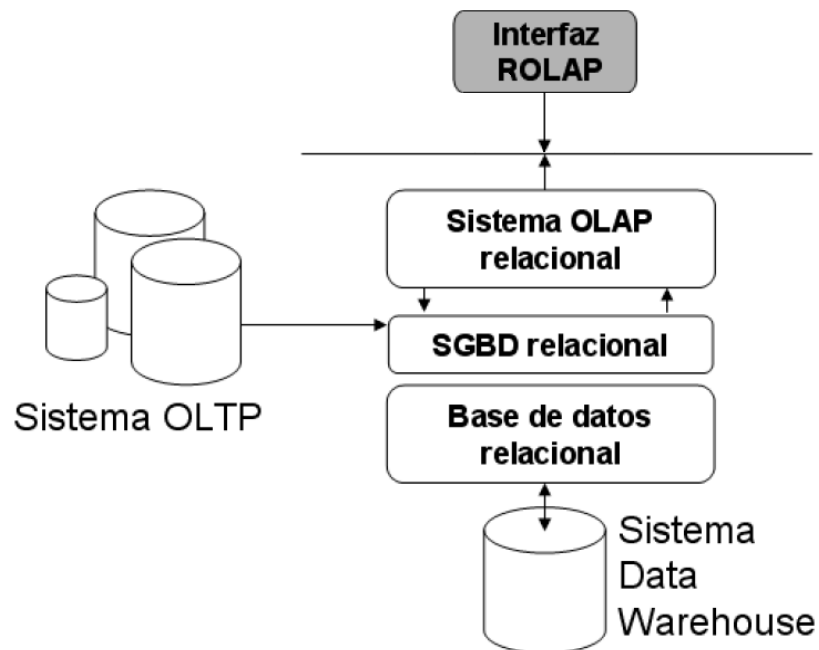
## 1.3 SISTEMAS ROLAP, MOLAP, HOLAP

### 1.3.1 ROLAP

La implementación de herramientas ROLAP débilmente acopladas con un SGBD se hace a través de SQL integrado en el lenguaje anfitrión del motor ROLAP. Los datos residen en el SGBD y son leídos registro por registro a través de ODBC, JDBC o de una interfaz de cursores SQL. La ventaja de esta arquitectura es su portabilidad. Sus principales desventajas son la escalabilidad y el rendimiento.

El problema de escalabilidad consiste en que las herramientas y aplicaciones bajo este tipo de arquitectura, cargan todo el conjunto de datos en memoria, lo que las limita para el manejo de grandes cantidades de datos. El bajo rendimiento se debe a que los registros son copiados uno por uno del espacio de direccionamiento de la base de datos al espacio de direccionamiento de la aplicación ROLAP. Estas operaciones de entrada/salida cuando se manejan grandes volúmenes de datos son bastante costosas, a pesar de la optimización de lectura por bloques presente en muchos SGBD (Oracle, DB2, Informix, PostgreSQL.) donde un bloque de tuplas puede ser leído al tiempo (figura 2.2).

Figura 2.2. Arquitectura ROLAP débilmente acoplada

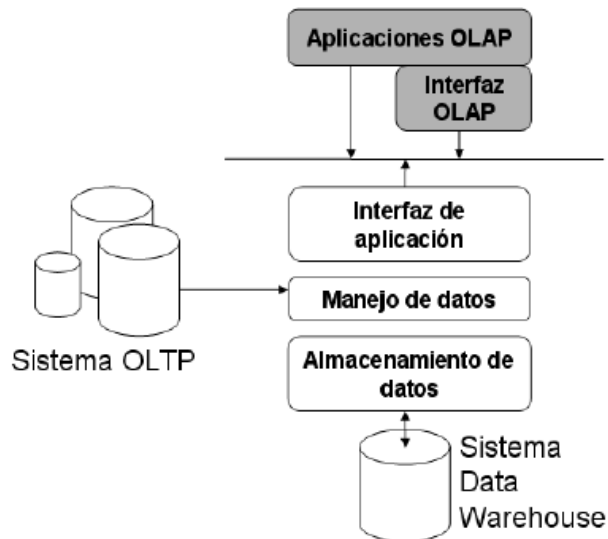


En este trabajo de investigación se trabajará con la arquitectura de un sistema ROLAP débilmente acoplada con un SGBD.

### 1.3.2 MOLAP

A diferencia de los sistemas ROLAP, el OLAP multidimensional es un modelo de datos de propósito especial, en cual las operaciones se hacen directamente sobre bases de datos multidimensionales (MDDs), que proporcionan un rendimiento muy alto en las consultas (figura 2.3). Otra ventaja del MOLAP respecto a ROLAP es la de tener operaciones nativas de OLAP, con un servidor de bases de datos multidimensional es fácil hacer pivot, roll-up, drill-down, con esto no es necesario recurrir uniones complejas y subconsultas del modelo relacional [1].

Figura 2.3. Arquitectura MOLAP



### 1.3.3 HOLAP.

OLAP híbrido, este sistema une las ventajas de los dos sistemas anteriores, en el siguiente sentido, se utilizaría un esquema MOLAP para almacenar las regiones más densas definidas en cada datacubo. Las dispersas se almacenarían en un esquema ROLAP.

Otra forma de realizar la hibridación sería, utilizar la estructura multidimensional como el cache de los datos del sistema de base de datos relacional. Con esto se busca responder rápidamente a las consultas OLAP con los datos en cache, en el caso de no encontrar los datos en cache, se construiría la consulta SQL para

recuperar los datos del sistema de base de datos relacional.

## 1.4 MODELO MULTIDIMENSIONAL

Cuando se creó el modelo relacional en los 80's, no se pensó en que sería utilizado para las aplicaciones de procesamiento analítico en línea, como reconoce su creador Codd en su artículo ([Cod93]), el modelo relacional cumple con las expectativas que despertó en su momento en lo concerniente al almacenamiento y acceso eficiente de los datos, pero no para el tipo de consultas y análisis de un sistema OLAP.

Es por esto que surgió este modelo de una necesidad básica para estos sistemas que es el procesamiento de gran cantidad de consultas en un tiempo más o menos corto, para que permitiera trabajar de forma interactiva con los usuarios. Desde que surgió este modelo se ha tratado de realizar una formalización del mismo lo que ha conllevado a que surja no solo una, sino varias formalizaciones del modelo multidimensional entre los cuales se tiene:

### 1.4.1 Estructura.

El modelo multidimensional está compuesto por dos elementos que son medidas o métricas y dimensiones:

#### **Medidas o hechos**

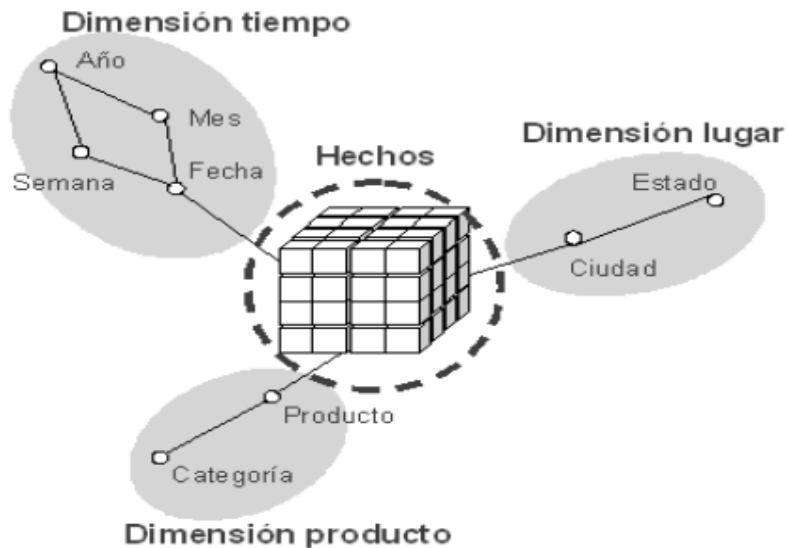
Contienen medidas del negocio que se está analizando: ventas, compras, producción, etc. Por lo general los datos que componen las medidas deben ser numéricos todo esto para que facilite las operaciones de agregación.

#### **Dimensiones**

Las dimensiones permiten definir un punto dentro del espacio multidimensional, además con ellas puedo elegir el nivel de detalle de las medidas que se esté analizando, todo esto debido a que cada dimensión puede tener uno o más niveles jerárquicos como ejemplo se tiene la figura 2.4.

En esta figura se puede observar tres dimensiones: tiempo, producto, lugar. Si se escoge la dimensión tiempo se puede tener los siguientes niveles: año, mes, semana, fecha; de igual forma, en la dimensión producto se tiene categoría y producto; en la dimensión lugar se encuentra estado y ciudad; de esta forma es posible moverse entre los diferentes valores de las jerarquías en las dimensiones, todo esto para ver información con mayor o menor detalle.

Figura 2.4. Muestra de la estructura multidimensional



### Operaciones de usuario

Con base a la estructura jerárquica de las dimensiones que se definieron, las operaciones que se puedan realizar sobre el modelo son: [LFD2003].

**Movimiento en la estructura jerárquica:** Para moverse en los diferentes niveles de jerarquía de una dimensión se definen las siguientes operaciones:

- **Roll-up:** con esta operación se logra subir en la jerarquía, esto significa, aumentar el tamaño del grano al que están definidos los hechos. Al aplicar esta operación se necesita resumir información para adaptar el nivel de detalle de los hechos. En este proceso de resumen se utilizarán operadores de agregación, en la figura 2.5 se puede observar el resultado de aplicar esta operación.

*Definición Algebraica:*

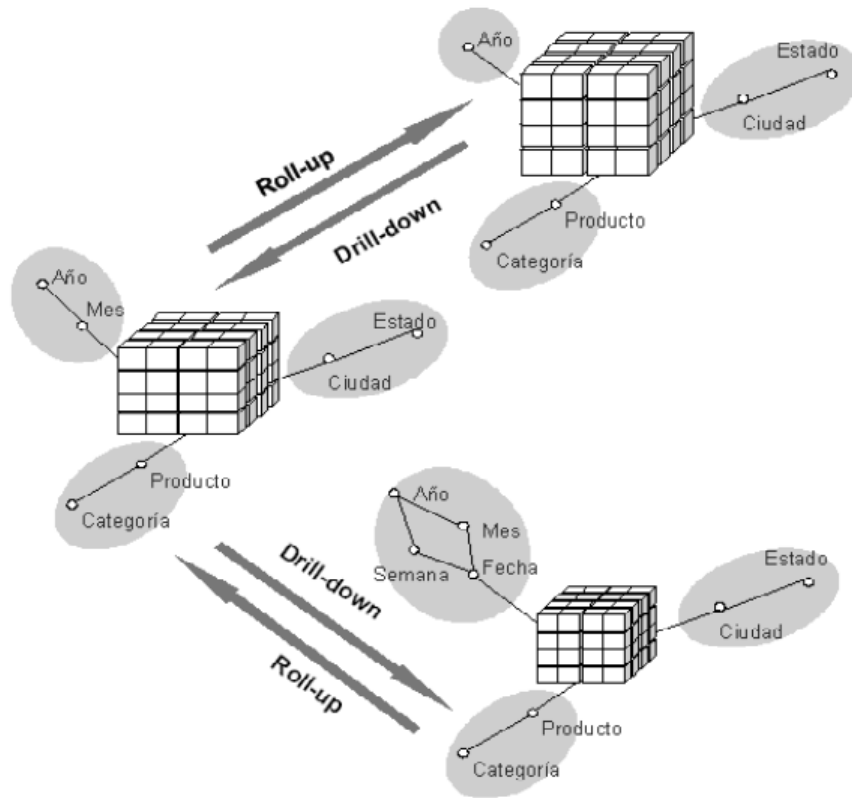
$$\uparrow (C_s^n, D_1, \dots, D_{n-1}) = C_{s'}^{n-1}$$

n: número de dimensiones del cubo C de nombre s.

D: dimensión.



Figura 2.5. Aplicación operaciones roll-up y drill-down sobre un cubo de datos.



- **Drill-down:** Esta operación es lo contrario de roll-up, ahora lo que se pretende es reducir el nivel de grano obteniendo un mayor nivel de detalle. Esto se traduce en cambiar el nivel de definición de los hechos a niveles inferiores de las jerarquías, en la figura 2.5 se puede observar el resultado de aplicar esta operación.

*Definición Algebraica:*

$$\Downarrow (C_s^n, C_{n+1}) = C_{s'}^{n+1}$$

n número de dimensiones del cubo C de nombre s.

**Operaciones de selección y proyección:** En algunos análisis puede que no se utilicen todos los valores (selección) de todas las dimensiones (proyección), las operaciones slice y dice cumplen con estas funcionalidades.

- **Slice:** Esta operación consiste en reducir la dimensionalidad del esquema eliminando alguna dimensión. Aplicar esta operación implica pérdida de detalle en los hechos (se aumenta la granularidad) por lo que se tendrá que utilizar operadores de agregación para la obtención de los nuevos, en la figura 2.6 se puede observar el resultado de aplicar esta operación.

*Definición Algebraica:*

$$\prod(C_s^n, D_1, \dots, D_m, D_{m+1} = d_{m+1}, \dots, D_n = d_n) = C_s^m \quad m \leq n$$

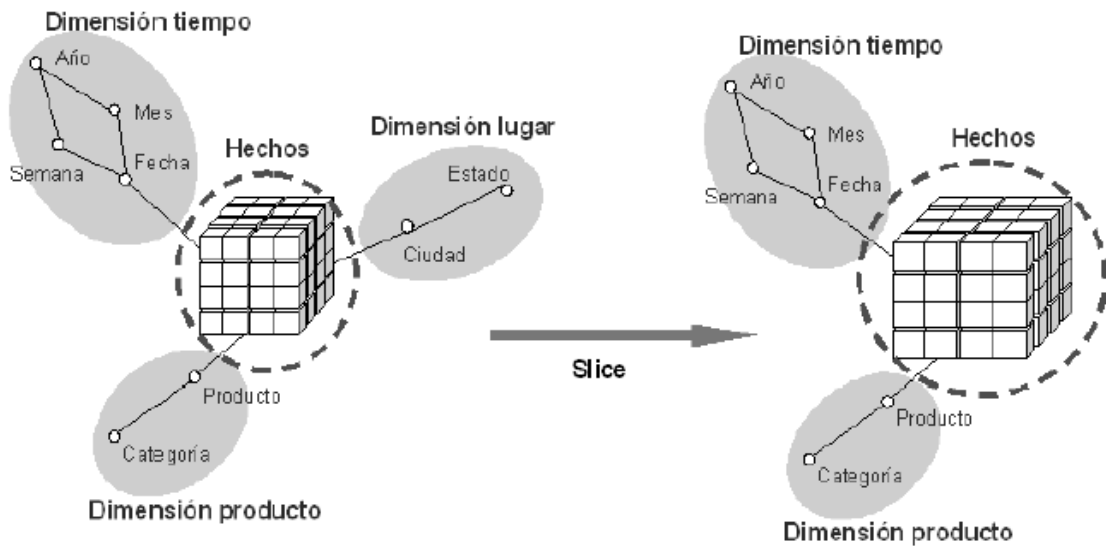
Sea  $C_s^n$  el cubo de n dimensiones y de nombre s.

D: las dimensiones.

d: las condiciones y dimensiones resultantes.

m: el número de dimensiones del cubo resultante.

Figura 2.6. Muestra de la aplicación de las operación slice.



- **Dice:** con esta operación lo que se hace es restringir los valores que se consideran en las dimensiones según alguna condición. No se modifica la estructura del cubo de datos en cuanto a las dimensiones y niveles de estas, sino restringiendo los valores que se considere. En este caso, no se modifica el

nivel de detalle de los hechos pero si su número, dado que aquellos que tuvieran como coordenadas valores no considerados deben ser eliminados.

*Definición Algebraica:*

$$\sigma(C_s^m, P) = C_s'^m$$

m número de dimensiones del cubo C de nombre s.  
p condición para la dimensión.

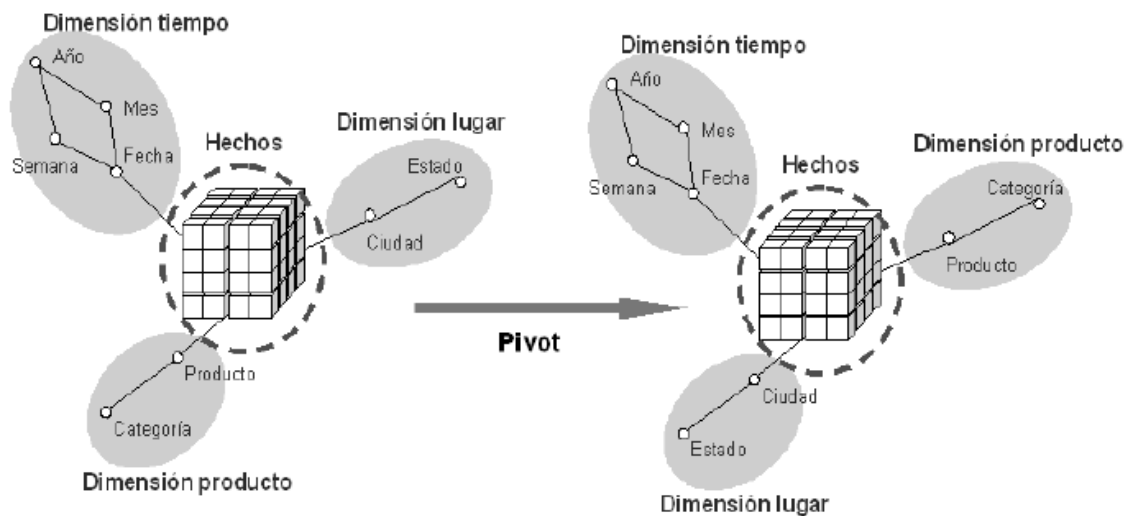
- **Operación de pivotaje:** Esta operación lo que persigue no es la modificación de la definición de la estructura de los cubos de datos. En realidad lo que se realiza es un cambio en el orden de las dimensiones, en la figura 2.7 se puede observar el resultado de aplicar esta operación.

*Definición Algebraica:*

$$\partial(C_s^n, D_1, \dots, D_n) = C_{s'}^n$$

Donde  $C_{s'}^n$  es un cubo con las mismas dimensiones del cubo  $C_s^n$  en distinto orden.

Figura 2.7. Muestra de la aplicación de las operación pivot.



## 2. ANÁLISIS DE HERRAMIENTAS OLAP

### 2.1 MONDRIAN PENTAHO™

Mondrian Pentaho versión 2.4.2 es un servidor OLAP escrito en Java. Con este servidor se puede analizar de una manera interactiva grandes juegos de datos almacenados en bases de datos sin escribir ninguna sentencia SQL [10].

Mondrian Pentaho versión 2.4.2, se caracteriza por tener un alto desempeño en el análisis de grandes o pequeños volúmenes de información, mediante la exploración de datos dimensional, por ejemplo análisis de ventas por región, producto, diferentes periodos de tiempo; aplicación de consultas con el lenguaje de expresiones multidimensionales (MDX) transformadas luego a SQL para obtener respuestas a consultas dimensionales, alto desempeño de consultas a través del uso de tablas agregadas en RDBMS, además de cálculos avanzados usando el cálculo de expresiones de el lenguaje MDX [17].

#### Requerimientos

- Requerimientos de Hardware  
Mínimo de memoria RAM libre de 128 MB.  
Procesador Pentium-500 MHz o superior.  
600 MB de espacio libre en disco.
- Requerimientos de Software  
Java SDK (1.4.2 o superior).  
Multiplataforma.  
Servidor web Tomcat (versión 5.0.25 o superior).

#### Instalación

La instalación y configuración básica del servidor Mondrian Pentaho es muy fácil ya que la distribución binaria de Mondrian Pentaho viene con una aplicación completa reconfigurada y los datos fuente vienen en una base de datos MS Access. Hay que seguir los siguientes pasos para configurar Mondrian Pentaho y ejecutarlo:

- *Instalar Java SDK* (versión 1.4.2 o superior) – se puede descargarlo de <http://java.sun.com/j2se/1.4.2/download.html>.
- *Instalar Servidor web Tomcat* (versión 5.0.25 o superior).

- *Descargar la versión binaria del servidor Mondrian Pentaho desde <http://mondrian.pentaho.org> y descomprímalo.*
- *Configuración de la fuente de los datos es imprescindible configurar la conexión a la base de datos correctamente para ejecutar y usar Mondrian Pentaho.*

La distribución binaria de Pentaho viene con una base de datos de prueba, que se llama *MondrianFoodMart.mdb* y está situada en la carpeta demo del archivo *Mondrian-xxx.zip*, esta base de datos se proporciona en dos formatos: como base de datos Microsoft Access (demo/access/MondrianFoodMart.mdb) y como un script SQL de sentencias de inserción (demo/FoodMartCreateData.sql), si se utiliza la base de datos de Access sólo es necesario definir una fuente de datos ODBC. Si se quiere probar a utilizar una fuente de datos que no sea Access, o si no está utilizando Windows, será necesario crear una base de datos vacía y cargar sobre la misma los datos proporcionado en el script, utilizando la utilidad MondrianFoodMartLoader.

- *Despliegue y ejecución se crea una carpeta en el directorio de publicación del servidor web Tomcat con el nombre “mondrian”, desde el paquete descargado con los binarios, se descomprime el archivo lib/mondrian.war sobre la carpeta anteriormente creada: /webapps/Mondrian, luego inicialice el servidor Tomcat, para finalizar escriba en su navegador la siguiente dirección <http://localhost:8080/mondrian>.*

## Manejo de la herramienta

El entorno principal de Mondrian Pentaho es el siguiente:

Mondrian examples:

- [JPivot pivot table](#)
- [JPivot pivot table by XMLA](#)
- [JPivot with 4 hierarchies](#)
- [JPivot with arrows](#)
- [JPivot with colors](#)
- [Various queries formatted using the Mondrian tag-library](#)
- [Basic interface for ad hoc queries](#)
- [XML for Analysis tester](#)

Other links:

- [Mondrian home page](#)
- [Mondrian project page](#)
- [JPivot home page](#)
- [JPivot project page](#)

En la pantalla principal se encuentra una serie de ejemplos, con los cuales se

puede analizar las diferentes operaciones factibles de hacer con el modelo multidimensional sobre la base de datos anteriormente configurada MondrianFoodMart.mdb.

## Test Query uses Mondrian OLAP




		Medidas		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
+All Media	+All Products	266.773	225.627,23	565.238,13

Slicer: [Year=1997]

[back to index](#)

Al escoger uno de los ejemplos se presenta en la parte superior el menú principal, el cual permite manipular los cubos mediante la aplicación de las operaciones (roll up, drill down, slice, dice), además de poder exportar los resultados de las consultas analíticas en línea; en la parte inferior se puede observar la tabla dinámica JPivot que es el cubo de datos inicial, para entender mejor el funcionamiento de la herramienta a continuación se detalla el menú principal:


La explicación del menú principal se realizará de izquierda a derecha:

- 
 Componente navegación permite agregar medidas, dimensiones y además realizar filtro sobre las dimensiones de la tabla dinámica JPivot.

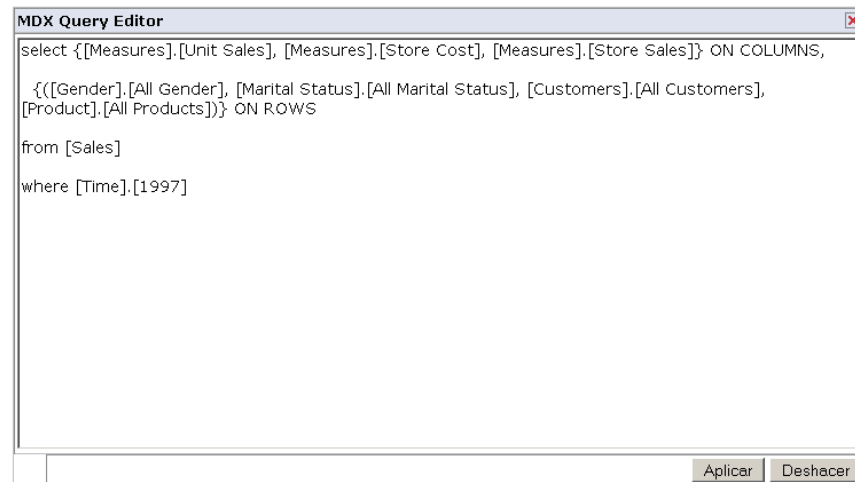


En la siguiente gráfica se observa que se realizó una operación dice sobre la dimensión Time la cual filtra los datos de tiempo a solo el año 1997.



- 
 Componente MDX que es el editor del lenguaje de expresiones multidimensionales, permite realizar consultas al modelo multidimensional y el resultado de esta consulta se ve reflejado automáticamente en la tabla dinámica JPivot.

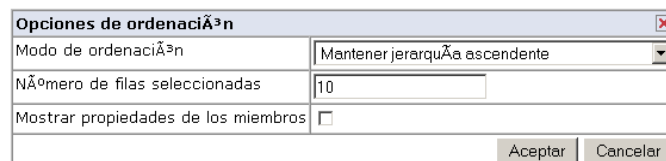
#### MDX Query Editor




				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
All Gender	All Marital Status	All Customers	All Products	266.773	225.627,23	565.238,13

Slicer: [Year=1997]

- 
 Componente de opciones de ordenación de la tabla.




- 
 Componente que muestra los padres, esto quiere decir los miembros que se encuentra más arriba en la jerarquía de las dimensiones.

Antes:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
+All Gender	-All Marital Status	+All Customers	+All Products	266.773	225.627,23	565.238,13
	M	+All Customers	+All Products	131.796	111.428,43	278.989,06
	S	+All Customers	+All Products	134.977	114.198,81	286.249,07

Después:



				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
(All)	(All)	(All)	(All)	266.773	225.627,23	565.238,13
+All Gender	-All Marital Status	+All Customers	+All Products	266.773	225.627,23	565.238,13
	All Marital Status	M	+All Customers	131.796	111.428,43	278.989,06
		S	+All Customers	134.977	114.198,81	286.249,07

- 
 Componente que permite ocultar repeticiones.  
 Antes:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
-All Gender	-All Marital Status	+All Customers	+All Products	266.773	225.627,23	565.238,13
-All Gender	M	+All Customers	+All Products	131.796	111.428,43	278.989,06
-All Gender	S	+All Customers	+All Products	134.977	114.198,81	286.249,07
F	-All Marital Status	+All Customers	+All Products	131.558	111.777,48	280.226,21
F	M	+All Customers	+All Products	65.336	55.439,34	139.195,80
F	S	+All Customers	+All Products	66.222	56.338,13	141.030,41
M	-All Marital Status	+All Customers	+All Products	135.215	113.849,75	285.011,92
M	M	+All Customers	+All Products	66.460	55.989,08	139.793,26
M	S	+All Customers	+All Products	68.755	57.860,67	145.218,66

Después:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
-All Gender	-All Marital Status	+All Customers	+All Products	266.773	225.627,23	565.238,13
	M	+All Customers	+All Products	131.796	111.428,43	278.989,06
	S	+All Customers	+All Products	134.977	114.198,81	286.249,07
F	-All Marital Status	+All Customers	+All Products	131.558	111.777,48	280.226,21
	M	+All Customers	+All Products	65.336	55.439,34	139.195,80
	S	+All Customers	+All Products	66.222	56.338,13	141.030,41
M	-All Marital Status	+All Customers	+All Products	135.215	113.849,75	285.011,92
	M	+All Customers	+All Products	66.460	55.989,08	139.793,26
	S	+All Customers	+All Products	68.755	57.860,67	145.218,66

- 
 Componente con el cual se muestran las propiedades.
- 
 Componente que permite suprimir filas y columnas vacías.


Antes:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
-All Gender	+All Marital Status	-All Customers	+All Products	266.773	225.627,23	565.238,13
		+Canada	+All Products			
		+Mexico	+All Products			
		+USA	+All Products	266.773	225.627,23	565.238,13
F	+All Marital Status	-All Customers	+All Products	131.558	111.777,48	280.226,21
		+Canada	+All Products			
		+Mexico	+All Products			
		+USA	+All Products	131.558	111.777,48	280.226,21
M	+All Marital Status	-All Customers	+All Products	135.215	113.849,75	285.011,92
		+Canada	+All Products			
		+Mexico	+All Products			
		+USA	+All Products	135.215	113.849,75	285.011,92



Después:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
-All Gender	+All Marital Status	-All Customers	+All Products	266.773	225.627,23	565.238,13
		+USA	+All Products	266.773	225.627,23	565.238,13
F	+All Marital Status	-All Customers	+All Products	131.558	111.777,48	280.226,21
		+USA	+All Products	131.558	111.777,48	280.226,21
M	+All Marital Status	-All Customers	+All Products	135.215	113.849,75	285.011,92
		+USA	+All Products	135.215	113.849,75	285.011,92


-  Componente que permite intercambiar los ejes lo que es igual a realizar la operación de pivotear.

Antes:

				Medidas		
Gender	Marital Status	Customers	Product	Unit Sales	Store Cost	Store Sales
+All Gender	+All Marital Status	-All Customers	+All Products	266.773	225.627,23	565.238,13
		+USA	+All Products	266.773	225.627,23	565.238,13

Después:

	Gender	
	+All Gender	
	Marital Status	
	+All Marital Status	
	Customers	
	-All Customers	+USA
	Product	Product
Medidas	+All Products	+All Products
Unit Sales	266.773	266.773
Store Cost	225.627,23	225.627,23
Store Sales	565.238,13	565.238,13


-  Componente detallar miembro con el cual se realiza la operación drill down y roll up en las jerarquías de las dimensiones a todos los miembros con el mismo nombre.

Antes:

		Medidas		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
-All Media	-Drink	24.597	19.477,23	48.836,21
	+Alcoholic Beverages	6.838	5.576,79	14.029,08
	+Beverages	13.573	11.069,53	27.748,53
	+Dairy	4.186	2.830,92	7.058,60
	+Food	191.940	163.270,72	409.035,59
	+Non-Consumable	50.236	42.879,28	107.366,33
Bulk Mail	-Drink	389	328,76	799,46
	+Alcoholic Beverages	100	90,51	223,65
	+Beverages	225	194,10	471,01
	+Dairy	64	44,15	104,80
	+Food	3.154	2.737,62	6.884,95
	+Non-Consumable	777	674,58	1.664,66
Cash Register Handout	-Drink	627	521,48	1.306,56
	+Alcoholic Beverages	189	157,54	397,58
	+Beverages	346	297,12	739,18
	+Dairy	92	66,83	169,80
	+Food	4.821	4.115,50	10.286,04
	+Non-Consumable	1.249	1.078,68	2.728,73

Después, realizando la acción detallar miembro con la operación roll up sobre la dimensión producto en su subnivel Drink, se obtiene lo siguiente:

Promotion Media	Product	Medidas		
		Unit Sales	Store Cost	Store Sales
-All Media	+Drink	24.597	19.477,23	48.836,21
	+Food	191.940	163.270,72	409.035,59
	+Non-Consumable	50.236	42.879,28	107.366,33
Bulk Mail	+Drink	389	328,76	799,46
	+Food	3.154	2.737,62	6.884,95
	+Non-Consumable	777	674,58	1.664,66
Cash Register Handout	+Drink	627	521,48	1.306,56
	+Food	4.821	4.115,50	10.286,04
	+Non-Consumable	1.249	1.078,68	2.728,73


-  Componente abrir detalle permite realizar la operación drill down y roll up en las jerarquías de las dimensiones solo y únicamente al miembro que se ha seleccionado.

Antes:

Promotion Media	Product	Medidas		
		Unit Sales	Store Cost	Store Sales
-All Media	+Drink	24.597	19.477,23	48.836,21
	+Food	191.940	163.270,72	409.035,59
	+Non-Consumable	50.236	42.879,28	107.366,33
Bulk Mail	+Drink	389	328,76	799,46
	+Food	3.154	2.737,62	6.884,95
	+Non-Consumable	777	674,58	1.664,66
Cash Register Handout	+Drink	627	521,48	1.306,56
	+Food	4.821	4.115,50	10.286,04
	+Non-Consumable	1.249	1.078,68	2.728,73
Daily Paper	+Drink	744	586,67	1.433,33
	+Food	5.536	4.765,06	11.960,65
	+Non-Consumable	1.458	1.207,49	3.085,83
Daily Paper, Radio	+Drink	683	514,66	1.297,60
	+Food	4.917	4.065,35	10.167,86
	+Non-Consumable	1.291	1.088,77	2.703,96
Daily Paper, Radio, TV	+Drink	884	675,40	1.707,03
	+Food	6.856	5.805,81	14.542,57
	+Non-Consumable	1.773	1.574,01	3.924,37

Después realizando la acción abrir detalle con la operación drill down sobre la dimensión producto en su subnivel Drink, se obtiene lo siguiente:

		Medidas		
Promotion Media	Product	Unit Sales	Store Cost	Store Sales
-All Media	-Drink	24.597	19.477,23	48.836,21
	+Alcoholic Beverages	6.838	5.576,79	14.029,08
	+Beverages	13.573	11.069,53	27.748,53
	+Dairy	4.186	2.830,92	7.058,60
	+Food	191.940	163.270,72	409.035,59
	+Non-Consumable	50.236	42.879,28	107.366,33
Bulk Mail	+Drink	389	328,76	799,46
	+Food	3.154	2.737,62	6.884,95
	+Non-Consumable	777	674,58	1.664,66
Cash Register Handout	+Drink	627	521,48	1.306,56
	+Food	4.821	4.115,50	10.286,04
	+Non-Consumable	1.249	1.078,68	2.728,73
Daily Paper	+Drink	744	586,67	1.433,33
	+Food	5.536	4.765,06	11.960,65
	+Non-Consumable	1.458	1.207,49	3.085,83
Daily Paper, Radio	+Drink	683	514,66	1.297,60
	+Food	4.917	4.065,35	10.167,86
	+Non-Consumable	1.291	1.088,77	2.703,96
Daily Paper, Radio, TV	+Drink	884	675,40	1.707,03
	+Food	6.856	5.805,81	14.542,57
	+Non-Consumable	1.773	1.574,01	3.924,37


- 
 Componente entrar en detalle realiza la operación drill down ocultando los elementos del nivel anterior.


Antes:

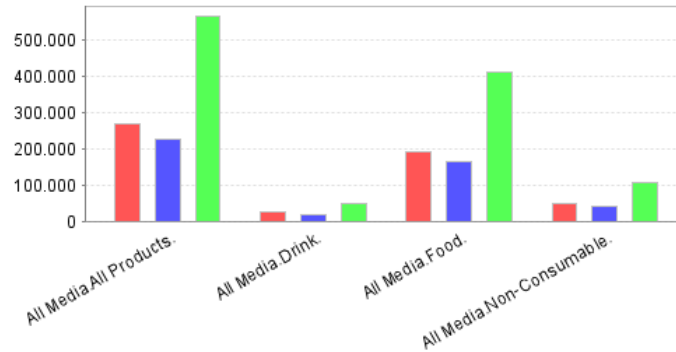
		Medidas		
Promotion Media	+Product	Unit Sales	Store Cost	Store Sales
+All Media	+Drink	24.597	19.477,23	48.836,21
	+Food	191.940	163.270,72	409.035,59
	+Non-Consumable	50.236	42.879,28	107.366,33

Después realizando la acción abrir detalle o dril down sobre la dimensión producto en su subnivel drink se obtiene lo siguiente:

		Medidas		
Promotion Media	+Product	Unit Sales	Store Cost	Store Sales
+All Media	+Alcoholic Beverages	6.838	5.576,79	14.029,08
	+Beverages	13.573	11.069,53	27.748,53
	+Dairy	4.186	2.830,92	7.058,60

- 
 Componente que permite visualizar los datos origen de la tabla dinámica JPivot.

- 
 Componente de visualización de forma gráfica de los datos contenidos en la tabla dinámica JPivot.







- 
 Componte que permite realizar configuración concerniente a las gráficas.

Chart Properties	
Chart Type	Vertical Bar
Enable Drill Through	<input type="checkbox"/>
Chart Title	
Chart Title Font	SansSerif Bold 18
Horizontal axis label	
Vertical axis label	
Axes Label Font	SansSerif Plain 12
Axes Tick Label font	SansSerif Plain 12 30°
Show Legend	<input checked="" type="checkbox"/> Bottom
Legend Font	SansSerif Plain 10
Show Slicer	<input checked="" type="checkbox"/> Bottom Left
Slicer Font	SansSerif Plain 12
Chart Height	300
Chart Width	500
Background (R, G, B)	255 255 255
OK Cancel	

- 
 Componente que permite configurar las opciones de impresión.

Print Properties	
Report Title	
Page Orientation	Portrait
Paper Size	A4
Custom Height/Width	29,70 cm 21,00 cm (0=default A4)
Table Width	<input type="checkbox"/> (off = auto) 0,00 cm
Chart on separate page	<input type="checkbox"/>
OK Cancel	

-  Componente que permite realizar reportes del contenido de la tabla dinámica JPivot, gráficas realizadas o datos de origen a un archivo Pdf de Adobe.
-  Componente que permite realizar reportes del contenido de la tabla dinámica JPivot, gráficas realizadas o datos de origen a un archivo de Microsoft office Excel.

## Ventajas

- Mondrian Pentaho versión 2.4.2 es una herramienta que facilita la manipulación de los cubos de datos, permitiendo realizar operaciones (slice, dice, drill down, roll up) sobre el modelo multidimensional, con tan solo unos pocos componentes de fácil utilización.
- La utilización de tablas agregadas sobre RDBMS permite un alto desempeño en los resultados de las consultas multidimensionales. análisis.
- Genera diferentes reportes estadísticos, con diferentes tipos de gráficas, permitiendo al usuario final escoger la que más se adecue al análisis que esté realizando.
- Permite trabajar con diferentes gestores de base de datos como son:
  - Apache Derby
  - Firebird
  - hsqldb
  - IBM DB2
  - Microsoft Access
  - Microsoft SQL Server
  - MySQL
  - Oracle
  - PostgreSQL
  - Sybase
- Permite exportar el resultado de las consultas multidimensionales además de las graficas realizadas sobre archivos en formatos pdf, xls.

## Desventajas

- La creación de los esquemas de configuración de los cubos es muy complicada de realizar sin la utilización de software adicional.
- Utilización de software adicional para la creación de los esquemas de los cubos como son (workbench, CubeDesigner).
- La instalación y configuración de esta herramienta son demasiado complicadas.

### 2.1.1 Creador de cubos, Pentaho™ Cube Designer v.0.7.2

Cube Designer es una herramienta software libre que ayuda o asiste al usuario en la creación de esquemas de los cubos Mondrian usando una interfaz gráfica de usuario, además permite, publicar el esquema del cubo creado en el motor de soluciones Pentaho.

#### Requerimientos

- JRE (versión 1.5 o superior).
- JDBC para la base de datos en la que se trabajara.
- Mondrian server/Pentaho server.
- S.O. Windows, Linux (Redhat/Suse), Mac OS.

La herramienta Cube Designer soporta cualquier driver JDBC. Los drivers JDBC para las bases de datos Oracle, MySQL, Microsoft SQL Server & Hibernate vienen incluidos con la distribución.

#### Instalación

Para descargar Cube Designer 0.7.2 se deberá ir a la dirección de internet [www.pentaho.org](http://www.pentaho.org).

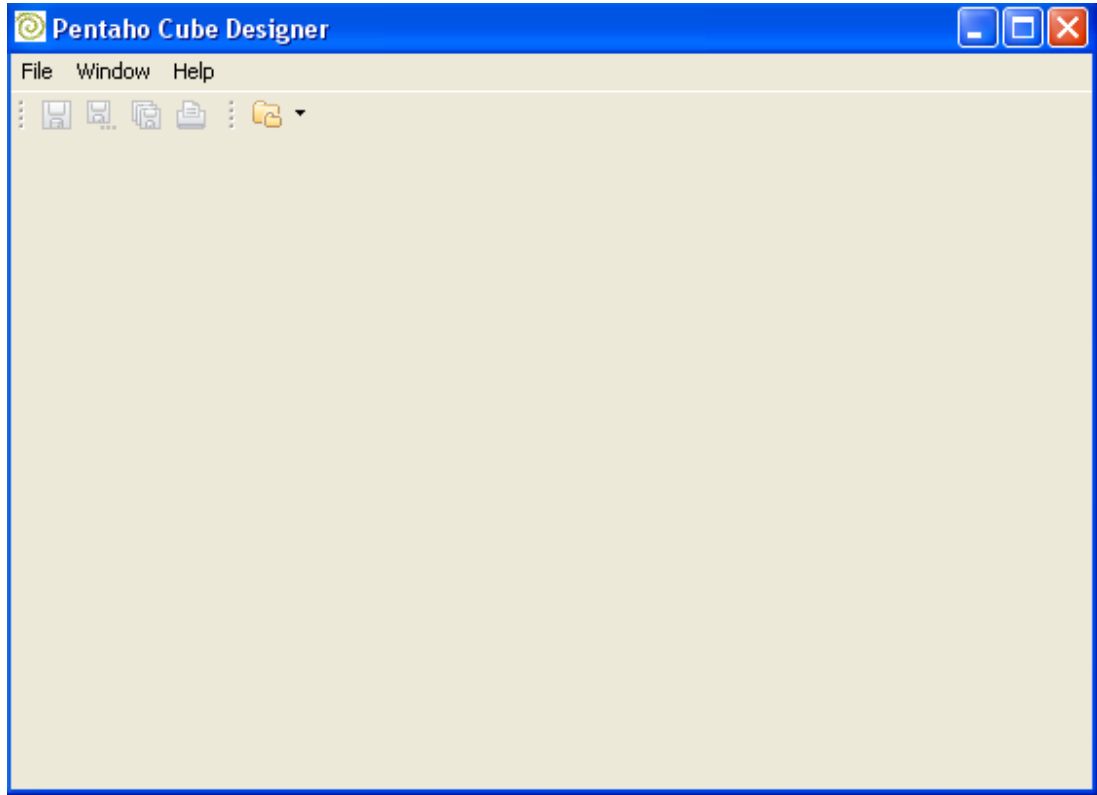
Para iniciar la aplicación solo basta con dar doble clic en "CubeDesigner.exe".

#### Manejo de la herramienta

En la siguiente imagen se observa la inicialización de la aplicación.



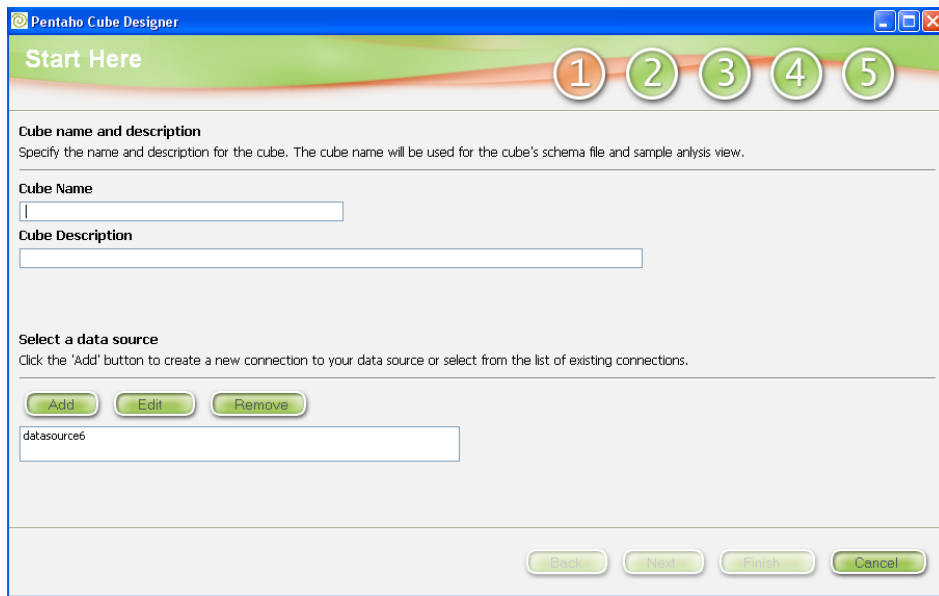
En esta imagen se observa el espacio de trabajo de la herramienta, para crear un esquema de un cubo se da clic en file, seguido a esto se da clic en la opción new cube schema.



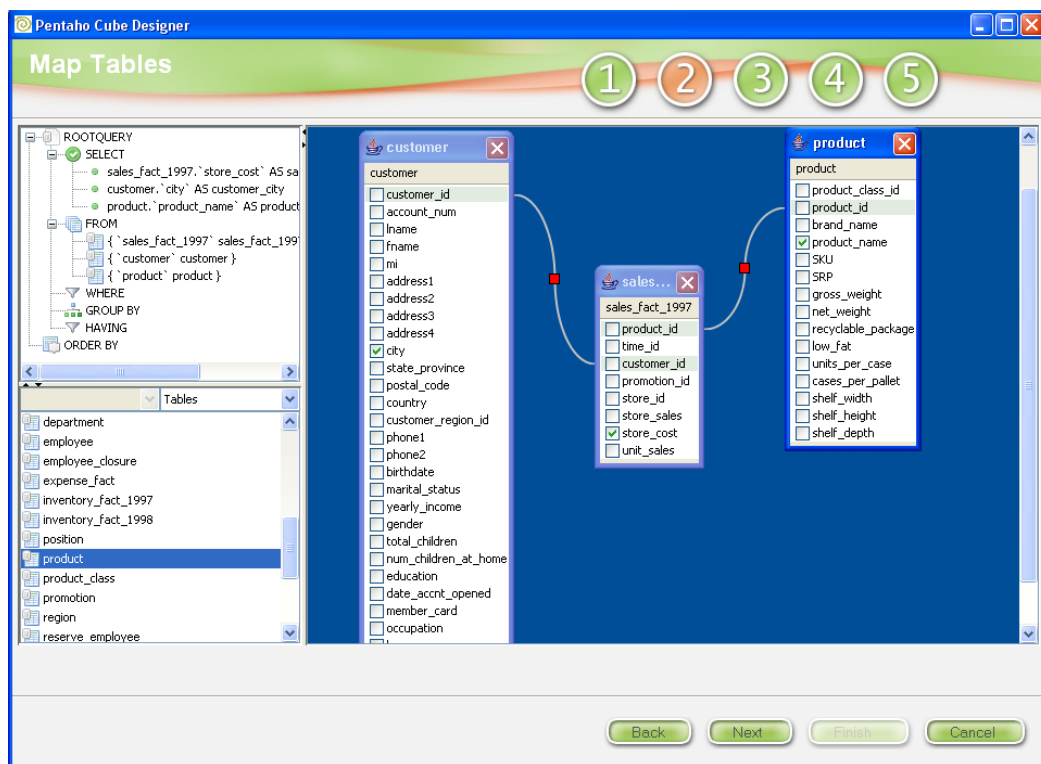
En esta ventana se puede observar que para realizar la definición de un esquema de un cubo solo se necesitan 5 pasos.

En el primer paso se puede configurar lo siguiente:

- Nombre del Cubo
- Descripción del Cubo
- Una fuente de datos



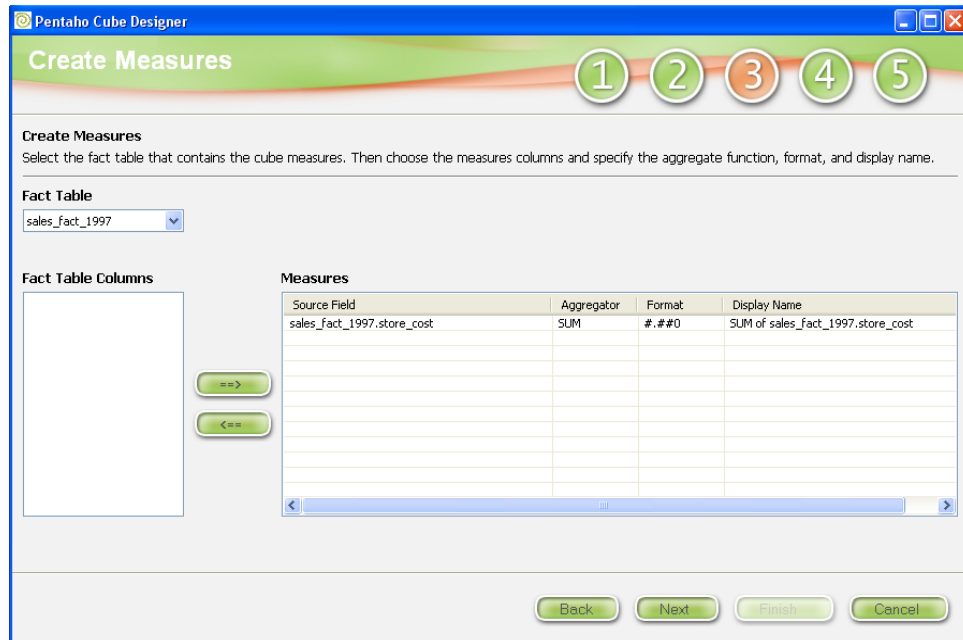
Luego de realizar las anteriores configuraciones, se observa una segunda ventana en donde se presenta las tablas de la fuente de datos anteriormente configurada. Aquí se observa el diseño multidimensional seleccionando tablas y campos del Data Warehouse.



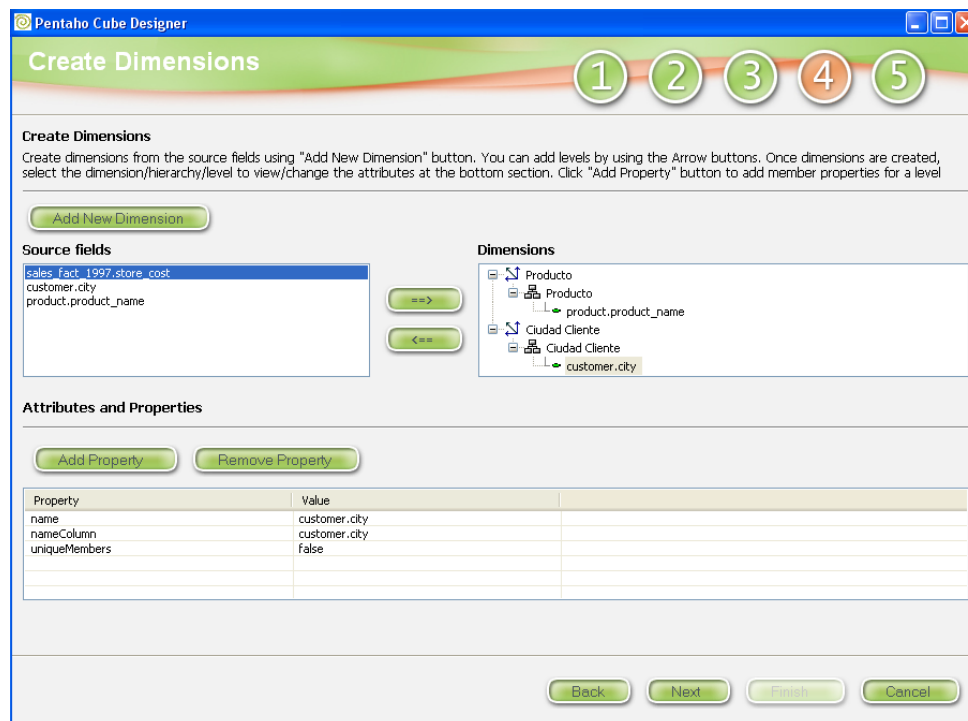
Después de seleccionar las tablas y los campos que se necesitan para el diseño



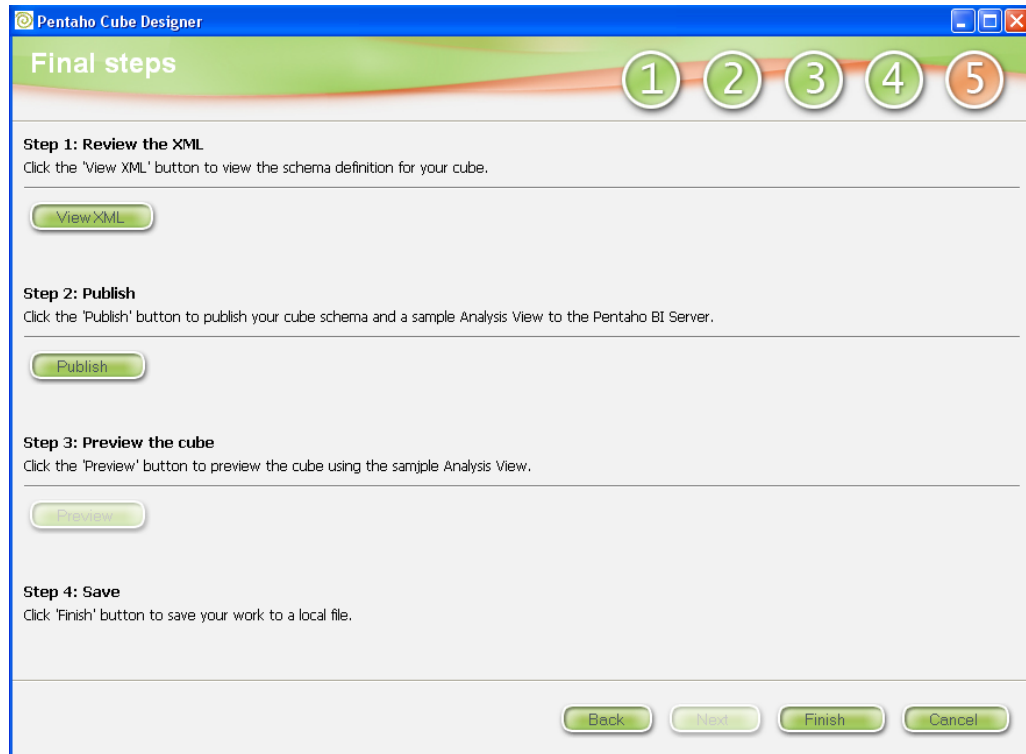
multidimensional se define cuál de estos campos son las medidas del negocio que se analizarán, además se define para cada medida la operación de agregación que se realizará sobre la misma (SUM, AVG, MIN, MAX, COUNT).



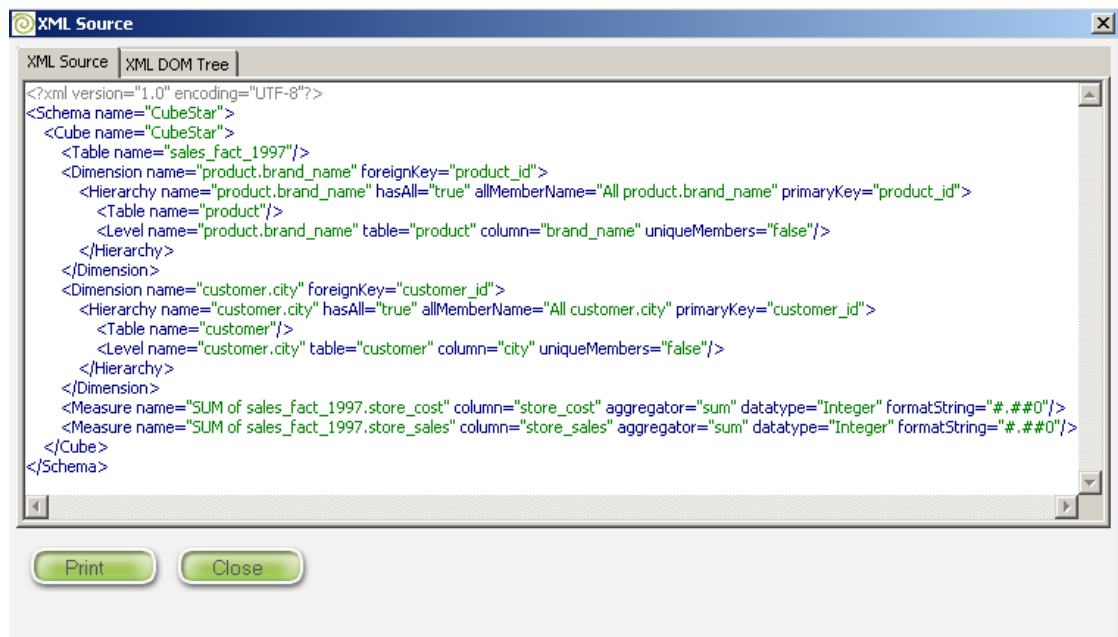
Luego de definir las medidas del negocio se pasa a crear las dimensiones y sus jerarquías.



En el quinto paso se puede realizar cuatro acciones:



La primera es visualizar la definición del esquema del cubo en formato XML o en XML DOM Tree.



La segunda es publicarlo en el Pentaho BI server, para realizar esto se debe digitar una clave de publicación, además la identificación y clave de acceso de un usuario del servido Pentaho BI.

Si el paso anterior fue exitoso se puede realizar un análisis sobre el cubo de datos. **stacd**



		Medidas
store.store_phone	warehouse.warehouse_country	SUM of inventory_fact_1997.units_ordered
+All store.store_phone	+All warehouse.warehouse_country	227,238227,238

Slicer:

© 2005-2007, Pentaho. Version: Pentaho BI Platform 1.6.0.GA.863



Como acción final se puede guardar la definición de esquema que se ha realizado.

## **Ventajas**

- Permite la creación de esquemas tipo estrella o sea una tabla de hechos y varias tablas de dimensiones.
- Permite la creación de esquemas tipo copo de nieve o sea una tabla de hechos y varias tablas de dimensiones jerarquizadas.
- El moldeamiento multidimensional es muy intuitivo debido al asistente que maneja la herramienta.

## **Desventajas**

- La instalación y configuración de esta herramienta es muy dispendiosa.

## **2.2 OLAP BROWSER**

El OLAP Browser es un software para el análisis de información cuando se necesita y como se necesita verla, ya sea desde un computador personal, una red local o desde la web.

Con OLAP Browser, el usuario puede trabajar minuciosamente los datos, realizar análisis gráficos y configurar e imprimir múltiples informes con tiempos de respuesta inmediatos [12].

## **Requerimientos**

- Requerimientos de Hardware
  - Mínimo de memoria RAM libre de 32MB.
  - Procesador mínimo Pentium III - 250 MHz o superior
  - 4 MB de espacio libre en disco.
- Requerimientos de Software
  - OLAP Browser se encuentra disponible para el Sistema Operativo Windows 98/2000/ME/NT/XP.
  - SQL Server Versions: mínimo SQL Server 7.0 or SQL Server 2000

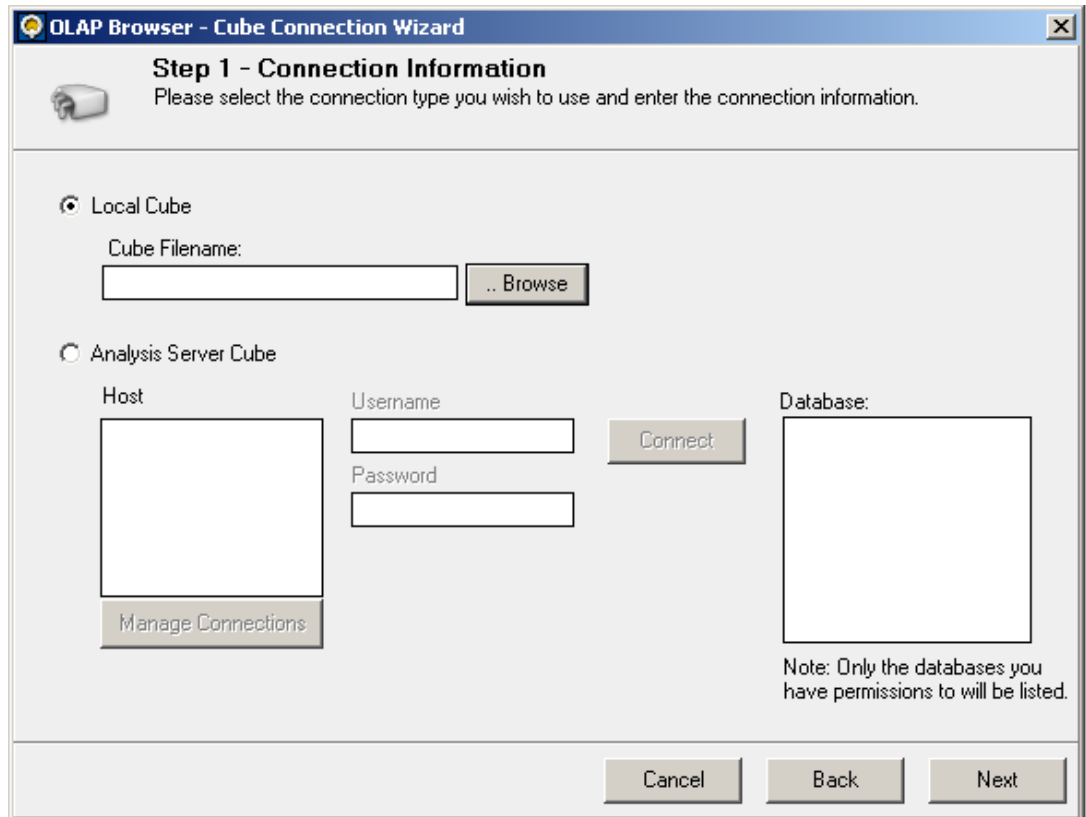
## **Instalación**

Para descargar la copia de evaluación de la versión OLAP Browser 3.0 se deberá ir a la dirección de internet [www.iqib.com](http://www.iqib.com).

El proceso de instalación es sencillo, basta con hacer clic en el botón de Next o Siguiente. Al final de este proceso aparece la ventana de instalación exitosa, la cual indica que toda la configuración de instalación se ha realizado satisfactoriamente y el OLAP Browser está listo para ser utilizado.

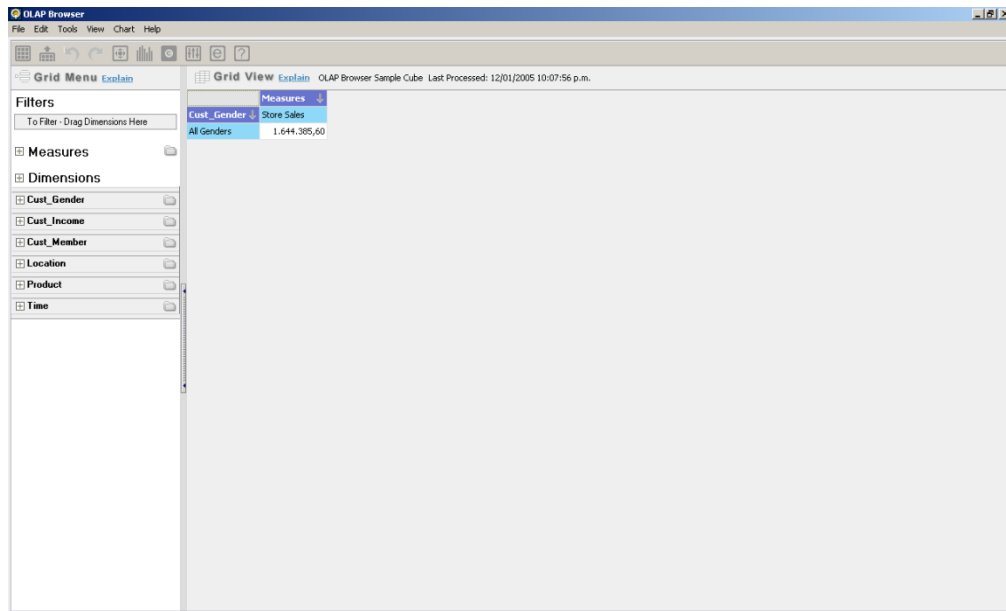
## Manejo de la herramienta

Antes de iniciar la aplicación, se presenta al usuario la opción de abrir un cubo local o un cubo existente en un host perteneciente a una red local.

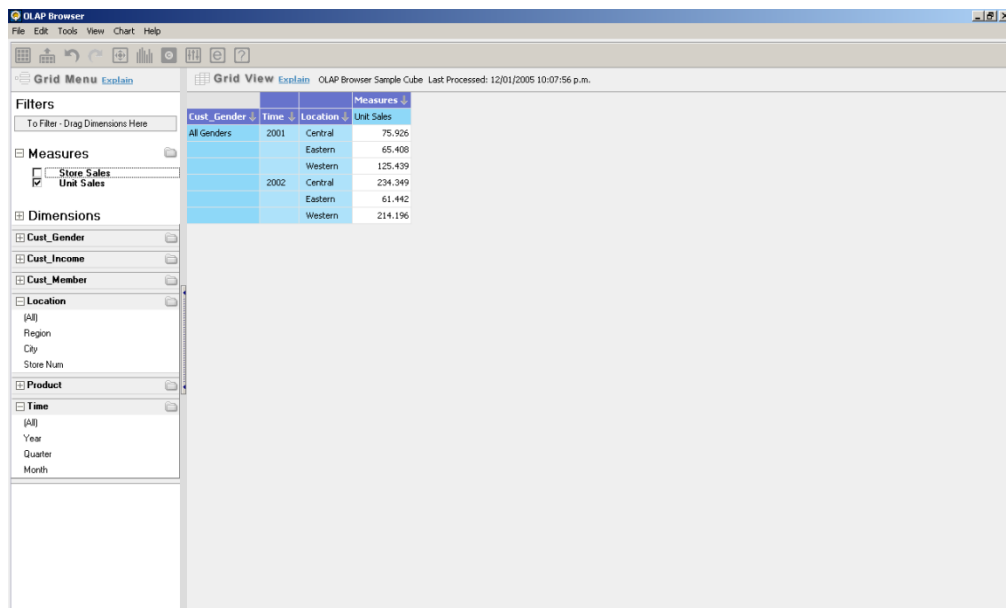


The screenshot shows a dialog box titled "OLAP Browser - Cube Connection Wizard" with a close button in the top right corner. The main heading is "Step 1 - Connection Information" with a sub-instruction: "Please select the connection type you wish to use and enter the connection information." There are two radio button options: "Local Cube" (selected) and "Analysis Server Cube". Under "Local Cube", there is a "Cube Filename:" label, an empty text box, and a ".. Browse" button. Under "Analysis Server Cube", there are four input fields: "Host" (a large empty box), "Username" (a text box), "Password" (a text box), and "Database:" (a large empty box). A "Connect" button is positioned between the Username and Password fields. A "Manage Connections" button is located below the Host field. At the bottom of the dialog, there are "Cancel", "Back", and "Next" buttons. A note at the bottom right states: "Note: Only the databases you have permissions to will be listed."


En el paso siguiente, se abrirá la ventana inicial en la cual se trabajará. La ventana se divide en 2 partes: La parte de la izquierda contiene las medidas a ser analizadas y sus correspondientes dimensiones, la parte de la derecha contiene el área de trabajo en el cuál se presentaran los datos que se están trabajando y las gráficas correspondientes.

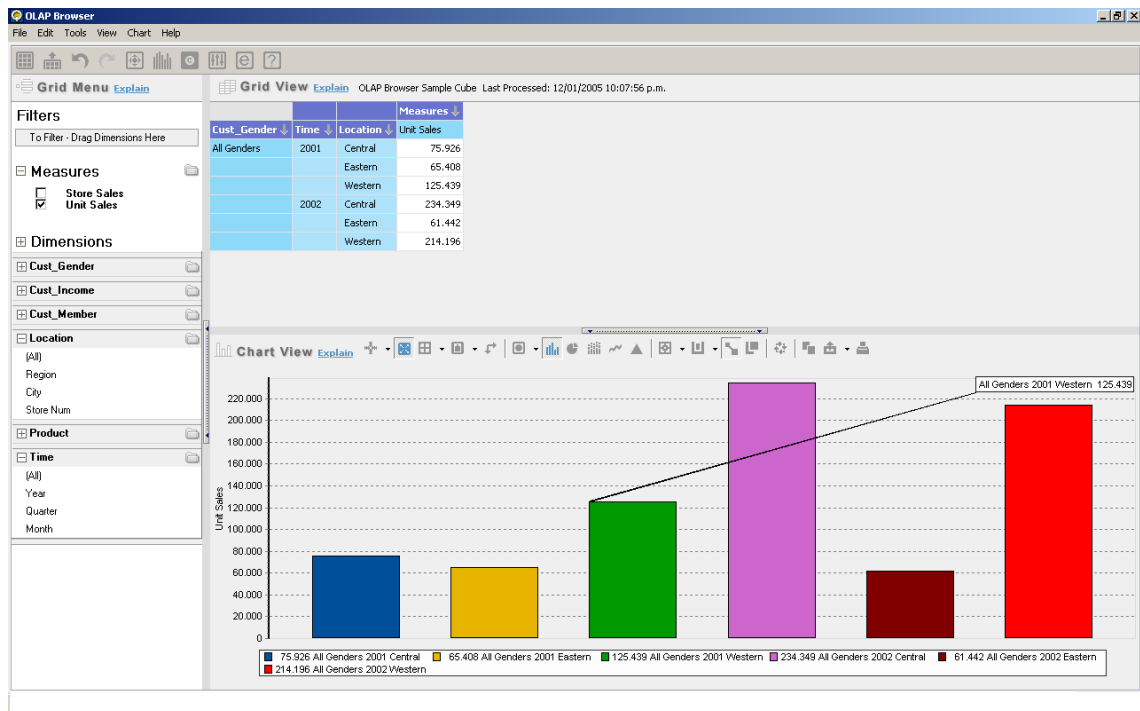


El usuario deberá ubicarse en la parte izquierda como primer acto, dar un clic en la opción Measures para determinar qué medidas desea analizar y seleccionar una de éstas, seguidamente se realiza el mismo procedimiento pero con la opción Dimensions que contiene las dimensiones a tener en cuenta en este análisis.



En la anterior imagen se indica que se ha seleccionado como medida a analizar Unit Sales y como dimensiones a Location, específicamente Region y Time, específicamente Year. Es decir, se va a analizar las unidades vendidas de cierto producto, este análisis se realizará por región y por año.

El siguiente paso es dar un clic en un botón que se ubica en la parte superior de la ventana llamado: “show chart view”  para generar la gráfica que corresponde a los datos que se están presentando.



En la imagen se observa que la parte derecha de la ventana se dividió en 2. En la parte superior se encuentran los datos y en la parte inferior la grafica de estos datos. La sección de la gráfica presenta diferentes opciones como agrupamiento, gráfica de pastel, gráfico lineal, entre otros que facilitan el adecuado análisis por parte del usuario.

### Características

- Permite exportar reportes con los siguientes formatos: HTML, PDF, RTF, EXCEL.

### Ventajas de la herramienta OLAP BROWSER

- Cuenta con una sencilla instalación que resulta muy intuitiva y rápida para el usuario, que puede o no, tener conocimiento de herramientas para la Inteligencia de Negocios.
- Su manipulación no resulta muy compleja, facilitando el uso de la herramienta.
- Posee la característica de poderse manipular a través de una red local y de la web.

- Ofrece garantía durante un periodo de tiempo después del cuál si el usuario no esta conforme con la herramienta se le devolverá su dinero.

### **Desventajas de la herramienta OLAP BROWSER**

- No se facilita mucho para pruebas ya que se proporciona al usuario que quiere probar la herramienta de 30 intentos de prueba (muy pocos). Además si un usuario por error abrió la herramienta, sin quererla usar e inmediatamente la cierra, ya ha gastado un intento de prueba sin siquiera haberla utilizado.
- No tiene una opción de guardar el cubo que se está trabajando en ese momento, lo que impide que se pueda continuar el análisis actual.
- Es software que se adquiere mediante licencias de uso lo que impide a las pequeñas y medianas empresas su acceso y su uso.
- Esta limitado al Sistema Gestor de Base de Datos SQL Server con las siguientes versiones: SQL Server Versions: SQL Server 7.0 or SQL Server 2003.

### 2.3 OLAPX v.4

#### **Descripción preliminar**

Es una sofisticada y completa herramienta para el análisis de información corporativa. Permite el análisis interactivo, reporte y presentación de cubos multidimensionales que se encuentren en bases de datos de Microsoft Analysis Services o en archivos locales.

Está diseñado para usuarios de cualquier negocio o nivel técnico para que puedan llevar a cabo el análisis de la información por sí mismos, crear reportes y consultas y compartírlas para mejorar el proceso de toma de decisiones de una compañía.

La suite de aplicaciones OLAPX versión 4 consta de:

- OLAPXApp: Aplicación de usuario final para consultar y analizar cubos multidimensionales.
- OLAPXACC: Un asistente para la creación de cubos.
- OLAPXDev: Herramienta para desarrollo de aplicaciones OLAP cliente-servidor.
- OLAPXWeb: Herramienta para servidor web.

OLAPXApp y OLAPXACC se consiguen con el mismo paquete de instalación, es decir no hay cargos extras.



## Requerimientos

- *Requerimientos de Hardware*

Mínimo de memoria RAM libre de 128MB.  
Procesador Pentium-500 MHz o superior  
50MB de espacio libre en disco.

- *Requerimientos de Software*

La herramienta OLAPX requiere que se instale previamente el componente “**Pivot Table Service Service Pack 3a**” en cada cliente que utiliza OLAPX. Normalmente se instala este servicio con Microsoft Office. OLAPX está disponible para plataformas desde Windows 2000 en adelante.

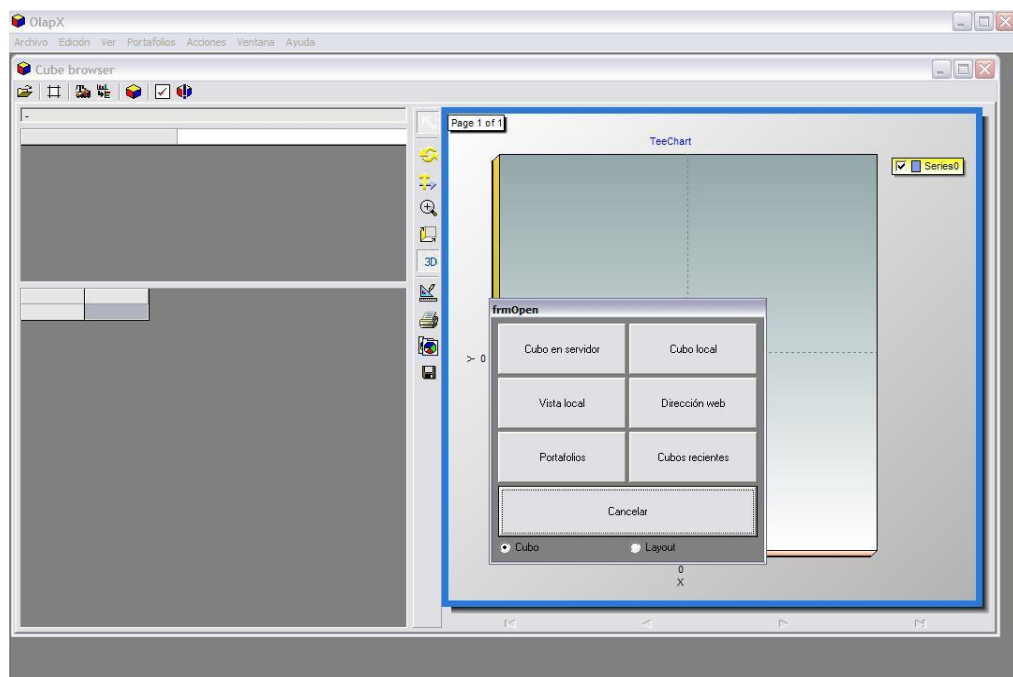
## Instalación

Para descargar la versión de evaluación de la aplicación OLAPX se debe ir a la página: <http://www.OLAPxsoftware.com/>.

El proceso de instalación es muy sencillo, basta con hacer clic en el botón de Next o Siguiente. Al final de este proceso aparece la ventana de instalación exitosa, la cual indica que toda la configuración de instalación se ha realizado satisfactoriamente y OLAPX está listo para ser utilizado.

## Manejo de la herramienta

El entorno principal al iniciar la aplicación OLAPX App es el siguiente:

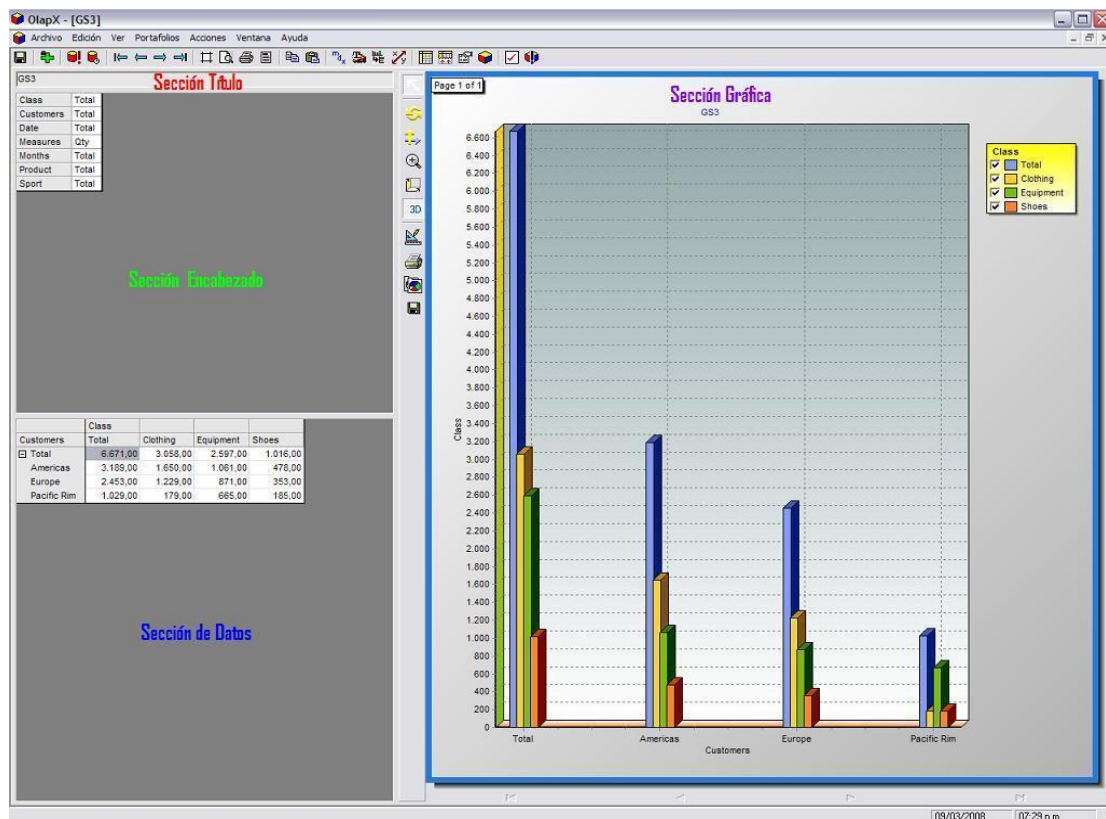


La aplicación en primera instancia pregunta al usuario la tarea que desea realizar:

- Utilizar cubo en servidor
- Utilizar cubo en archivos o redes locales
- Vistas locales de cubos
- Portafolios de reportes
- Abrir archivos desde direcciones URL

Realizada la elección el entorno se mostrará de la siguiente manera:

La pantalla se divide en cuatro secciones:



### Sección de título

Título del cubo con el que se trabaja. Normalmente aquí se da el nombre al cubo.

### Sección de encabezados

Aquí aparecerán todas las dimensiones que no estén en área de datos. De estas dimensiones solo se puede escoger un elemento. El valor de este elemento aparecerá a la derecha del nombre de la dimensión. Estos valores se utilizarán como criterio de búsqueda.

## Sección de datos

En esta sección aparecen los datos de una dimensión cruzados con otra, o también varias dimensiones cruzadas con otras. En la figura anterior por ejemplo se muestra el total de ventas cruzado contra las dimensiones que aparecen aquí: *Class* y *Customers*. Para intercambiar una dimensión del encabezado por una del área de datos, solo se hace clic en el valor del encabezado deseado y se da otro clic en un campo de alguna dimensión en la sección de datos.

Otras operaciones que se pueden realizar son:

- **Pivote**

Un pivote ocurre cuando se quiere intercambiar la posición de las dimensiones en los ejes de coordenadas. Por ejemplo si la dimensión *Date* está en el eje vertical del área de datos y la dimensión *Class* en el horizontal, con la instrucción de pivote, se intercambia su posición.

Se puede hacer de dos formas: la primera dando un clic en la esquina superior derecha de la tabla de la sección de datos y la segunda con la opción del menú **Acciones -> Cambiar eje XY** o el botón correspondiente en la barra de herramientas.

- **Drill-down**

Las dimensiones de los cubos son jerárquicas por naturaleza. No siempre, pero normalmente todas las dimensiones tienen un campo de grano total, subtotales de primer nivel, de segundo nivel y así hasta llegar al último nivel de detalle definido. La operación de *drilldown* le permite abrir un nuevo nivel de detalle en una dimensión.

Para hacer uso de esta opción tan sólo se hace un clic en cualquier nivel de la dimensión de la sección de datos para desplegar sus subniveles.

- **Agregado de dimensiones**

De la misma forma en que se intercambia las dimensiones, se puede agregar dimensiones al área de datos desde la sección de encabezado presionando la tecla de Control cuando se da clic en el área de datos.

## Sección de gráfica

La gráfica es la encargada de representar visualmente y en determinado tipo de diagrama los valores que se encuentran en la sección de datos.

El formato de la gráfica se puede modificar fácilmente pues es un elemento muy versátil. Con cualquier botón en su barra vertical de opciones se puede manejarla. De esta manera, la gráfica puede ser: girada (3D), cambiada de posición, aumentar la profundidad en ella, observarla únicamente en 2D, realizar zoom, cambiar el tipo de diagrama que se está manejando, imprimirla, hacer una copia de ella al portapapeles o guardarla.

### **Opciones de formato**

Este tipo de opciones servirán para dar formato a los valores numéricos en cuanto a orden y visualización se refiere: color de las celdas (bordes, relleno,etc), fuente de los textos, formatos numéricos y estilos (aplanado, 3D y 3D ligero). Para esto se va al menú Edición -> Opciones de formato.

- *Formato*

Este diálogo permitirá modificar el formato de los datos del cubo.

Se pueden cambiar los colores de distintas áreas de las matrices. Los datos son los valores en la matriz, los títulos son los elementos de las dimensiones del cubo.

- *Valores*

Esta sección se utiliza para dar formato a los datos dependiendo de los valores del cubo. Se puede aplicar formato condicional al cubo, ya sea desde valores absolutos o como un porcentaje superior o inferior al valor de una fórmula aplicada a los datos mostrados. Estas fórmulas pueden ser promedio, media o desviación estándar.

Los botones Fondo y Frente se utilizan para cambiar el color de fondo y letra de los valores que se encuentren arriba, abajo e entre los dos valores dados. Por ejemplo se puede pintar con un color de fondo rojo todos los valores que se encuentren un 15% por debajo del promedio de los valores mostrados.

### **Opciones de dimensión**

Aquí se puede mirar cómo escoger qué valores se observarán. Algunas de estas opciones aplican solo a dimensiones que se encuentran en el área de datos, ya sea vertical u horizontal.

Es posible cambiar la vista de una dimensión haciendo clic derecho en un campo de la dimensión y escogiendo una opción del menú **Vista de dimensión**.

Aquí se puede escoger formas para ver los miembros de una dimensión, como *jerarquía completa* o *nodos finales* o sólo mostrar el campo al que se dio clic (*sólo yo*). También se puede escoger la vista *Explorador* que es la vista predeterminada o *sólo mis hijos* que muestra el valor con sus miembros directos.

En el diálogo *opciones de dimensión* se darán más opciones. Está disponible dando doble clic en el área de encabezados o con la opción del menú **Edición -> Opciones de dimensión** o dando clic en el botón correspondiente de la barra de herramientas.

- **Elementos**

Aquí se escogerá qué elementos de las dimensiones se verán y de qué manera. Normalmente siempre se estará en modo de *Explorador*. Esta opción va a abrir más detalle para cada miembro de la dimensión. Hay tres maneras de ver una dimensión: como *jerarquía*, como una *lista* o como *un solo valor*.

Una jerarquía muestra los miembros de una dimensión de distintos niveles, como por ejemplo el continente, país y compañía. Este modo se selecciona con los tipos *Explorador*, *jerarquía de N niveles* y *jerarquía completa*.

Una lista de campos se muestra cuando todos los miembros son de la misma dimensión sin importar su nivel en la jerarquía. Aquí se selecciona la *lista de campos*, *nodos finales*, *sólo mis hijos* o *campos de un nivel* para ver listas de campos.

El tercer tipo es cuando sólo se quiere ver un elemento de una dimensión en el cubo. Para este modo se utiliza la opción *sólo yo*.

- **Propiedades**

Con cubos de *MS Analysis* [1], se puede guardar atributos de miembros de una dimensión junto con los miembros. Por ejemplo, si se está viendo un cliente, se podría ver su dirección, teléfono, etc. Esta pestaña permite ver qué atributos se van a ver junto con los miembros.

**[1] Microsoft SQL Server Analysis Services**

*Microsoft SQL Server Analysis Services (SSAS) ofrece funciones de procesamiento analítico en línea (OLAP) y minería de datos para aplicaciones de Business Intelligence. Analysis Services admite OLAP y permite diseñar, crear y administrar estructuras multidimensionales que contienen datos agregados desde otros*

*orígenes de datos, como bases de datos relacionales. En el caso de las aplicaciones de minería de datos, Analysis Services permite diseñar, crear y visualizar modelos de minería de datos que se construyen a partir de otros orígenes de datos mediante el uso de una gran variedad de algoritmos de minería de datos estándar del sector.*

- **Valores**

Cuando los campos se listan en una lista o en modo de un solo valor, es posible desplegarlos junto con algunas fórmulas de valores calculados. Se puede desplegar listas de campos junto con su *total, promedio, mínimo, máximo, número de registros, media o desviación estándar*. Para esto se va a la pestaña Valores y se escoge las funciones total y promedio.

- **Campos calculados**

Muy a menudo se necesitará crear nuevos campos y analizarlos desde la perspectiva de su valor. Por ejemplo, si se quiere analizar crecimiento entre dos periodos de tiempo en vez de ver valores anuales. Se ofrece la funcionalidad de *MDX\** para crear nuevos campos.

\* **MDX** es el lenguaje estándar de Microsoft para el acceso a cubos multidimensionales. Es un lenguaje robusto que le permite crear desde simples expresiones hasta complejas fórmulas.

## **Ventajas**

- Se puede realizar una manipulación realmente fácil de los siguientes elementos:
  - Tamaño de los elementos de la ventana.
  - Dimensiones del cubo.
  - Tablas de datos
  - Gráfica y sus modificaciones
- Las operaciones de drill-down y drill-up se hacen de manera muy sencilla, tan sólo navegando en la sección de datos.
- La operación de pivote es bastante sencilla de realizar, tan sólo con dar un clic en la parte superior izquierda de la tabla ubicada en la sección de datos.
- Se puede navegar hacia cualquier parte de nuestro trabajo desarrollado, ya sea hacia adelante o hacia atrás, con tan sólo dar un clic en uno de los botones “Ir a pantalla anterior/siguiente”, o ir al inicio o final con uno de los botones “Ir a la primera/última pantalla”.

- OLAPX lleva un archivo de bitácora de errores llamado OLAPXApp.log que puede ser utilizado por los distribuidores del producto para soporte técnico.
- El lenguaje MDX es un lenguaje muy versátil que proporciona muchas funcionalidades a la hora de trabajar con el modelo multidimensional.

### **Desventajas**

- Las herramientas OLAPX vienen de forma separada en distintas aplicaciones.
- OLAPX es una herramienta de software propietario, lo que limita a las empresas de mediano o bajo perfil adquirir este tipo de aplicaciones, ya que sus licencias son costosas (desde los US \$99 hasta los US \$1.600 según el número de licencias).

#### 2.3.1 Creador de cubos OLAPx Acc

### **Descripción**

El asistente para la creación de cubos es una herramienta creada por OLAPX Software con el propósito de brindar soporte a usuarios de cualquier nivel técnico que tengan necesidad de analizar información de su empresa, pero aunque tienen acceso a los datos fuente, no tienen la información en formato multidimensional.

OLAPXACC una herramienta que permite tomar datos fuente y convertirlos en cubos multidimensionales, los cuales pueden ser aprovechados posteriormente por otras herramientas como Microsoft Excel y otras herramientas que accedan a cubos multidimensionales que se encuentren en bases de datos de Microsoft Analysis Services o en archivos locales.

### **Requerimientos**

Los mismos que se requieren para OLAPX, ya que esta herramienta se instala en conjunto con la aplicación OLAPXApp.

### **Instalación**

Esta actividad se lleva a cabo cuando se instala la aplicación OLAPXApp ya que estas dos aplicaciones se consiguen en un mismo paquete.

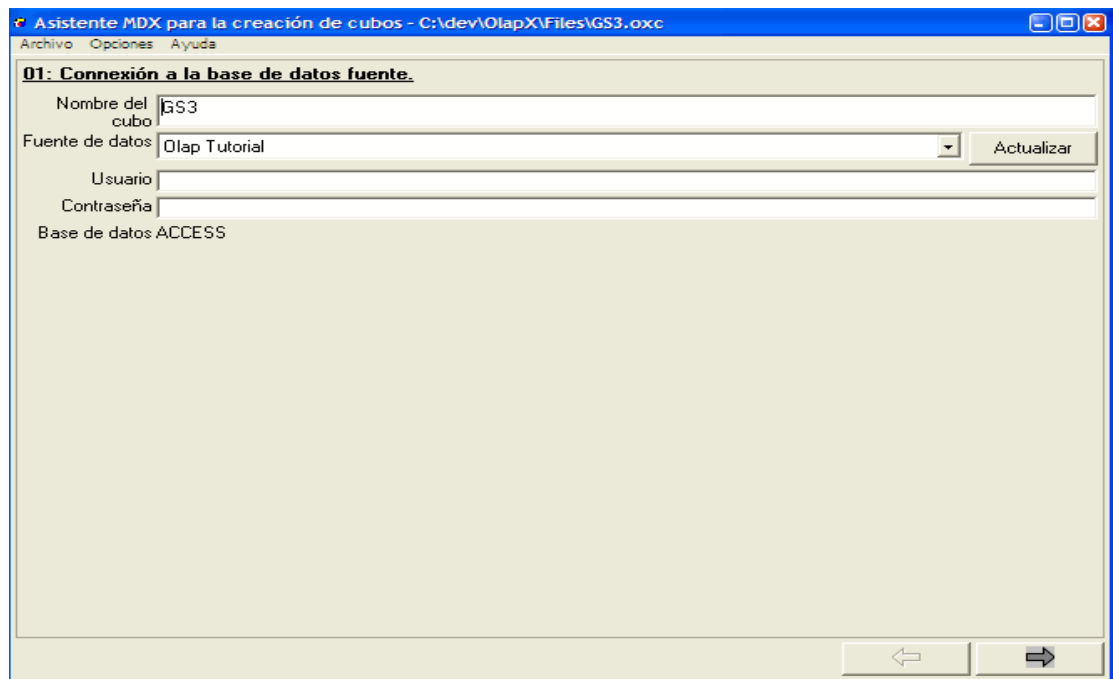
### **Manejo de la herramienta**

A continuación se resumirán los pasos más importantes para la construcción de un cubo de datos.

## Conexión a la base de datos

Como primera instancia del asistente se tiene una ventana en la cual se va a definir la base de datos con la cual se va a comunicar para traer los datos que van a formar parte del cubo. Se debe tener en cuenta las siguientes condiciones:

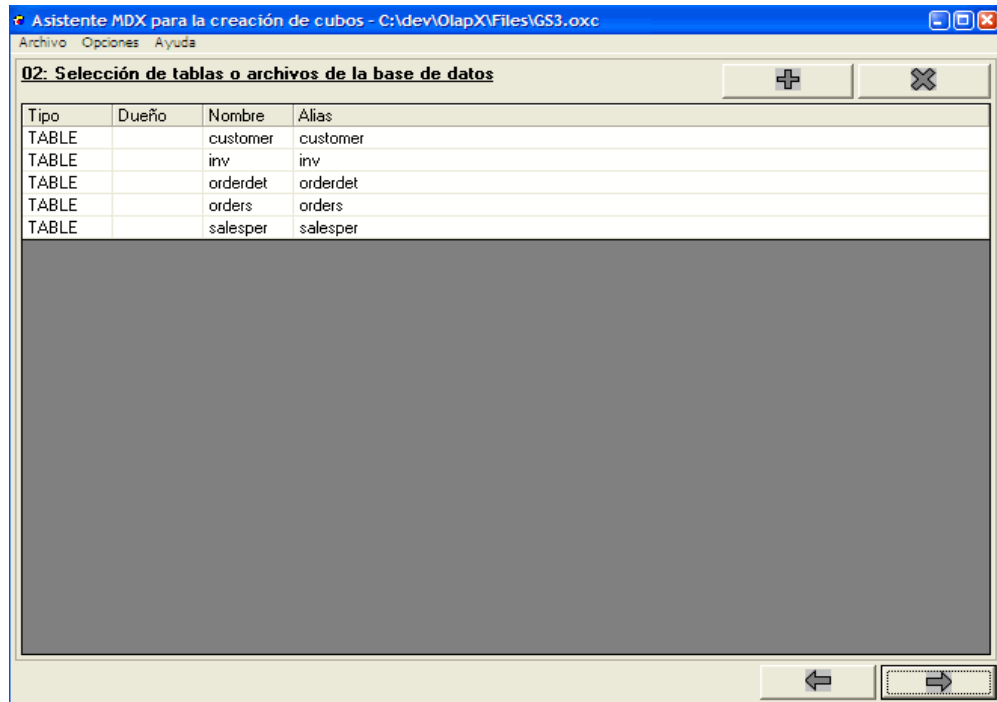
- **Nombre del cubo**
- **Fuente de datos:** Esta es una conexión *ODBC* que se utiliza para conectarse a una base de datos. Aquí se escoge una conexión ODBC de la lista o se puede utilizar el Administrador de ODBC de Windows para crear una nueva fuente de datos.
- **Actualizar:** Actualiza la lista de fuentes de bases de datos disponibles.
- **Usuario y contraseña:** Si la conexión a la base de datos requiere de un nombre de usuario y/o contraseña.
- **Base de datos:** Esta es una etiqueta que indica el tipo de base de datos que se va a utilizar para crear el cubo. Depende de la conexión a la base de datos y no se puede modificar.



## Selección de tablas o archivos de la base de datos

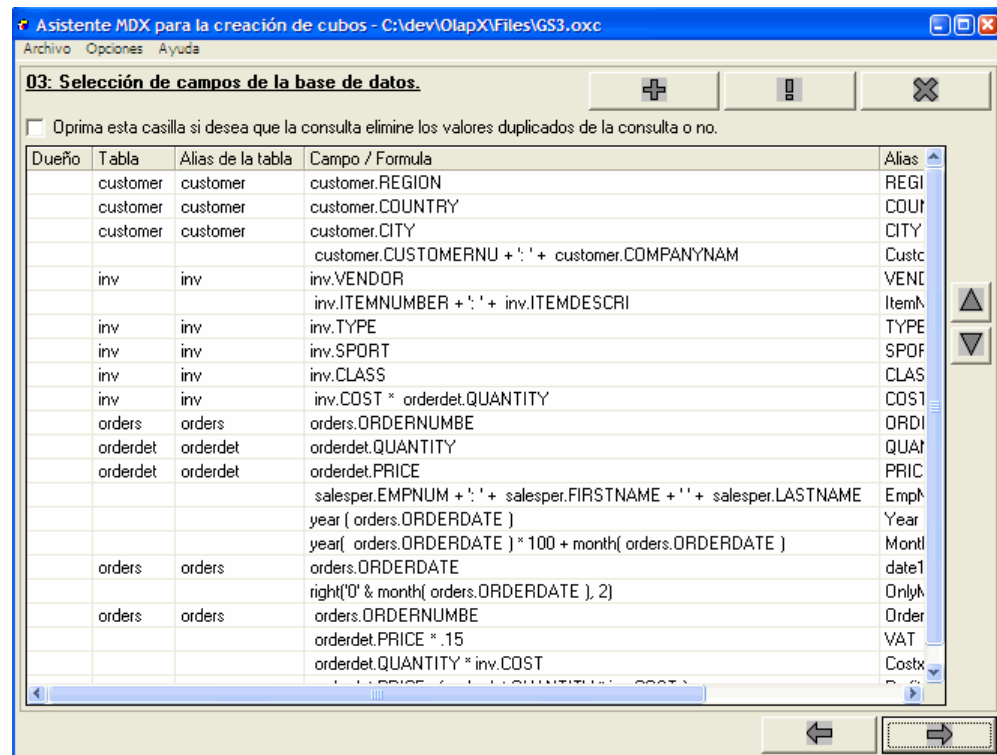
Este paso consiste en definir y escoger las tablas o los archivos de la base de datos en donde se encuentran los datos que se van a utilizar (hechos y dimensiones).





### Selección de campos de la base de datos.

Este paso se utiliza para escoger los campos de las tablas (seleccionadas en la sección anterior) que se van a utilizar durante la creación del cubo. Cuando se entra a esta sección, se escogen todos los campos de todas las tablas que se escogieron en la sección anterior.



Se puede observar que aparecen unas opciones diferentes a las vistas en los pasos anteriores:

- **Oprima esta casilla si desea que la consulta elimine los valores duplicados de la consulta o no:** En las bases de datos normalmente se guardan datos no duplicados. En un negocio no se querrá guardar un misma factura dos veces, porque aparecería dos veces en un listado de cuentas por cobrar más dinero del que se va a poder cobrar, o que se va a ganar más dinero del que se va a ganar. Pero hay consultas en las que se pueden traer datos duplicados, por ejemplo, si se quiere ver las ciudades en las que se ha vendido productos. Obviamente va a ver ciudades en las que se ha vendido más de una vez, por lo que saldrían duplicadas en la consulta.
- **Editar**
- **Flechas arriba y abajo:** Permite cambiar el orden de los campos dentro de la consulta.

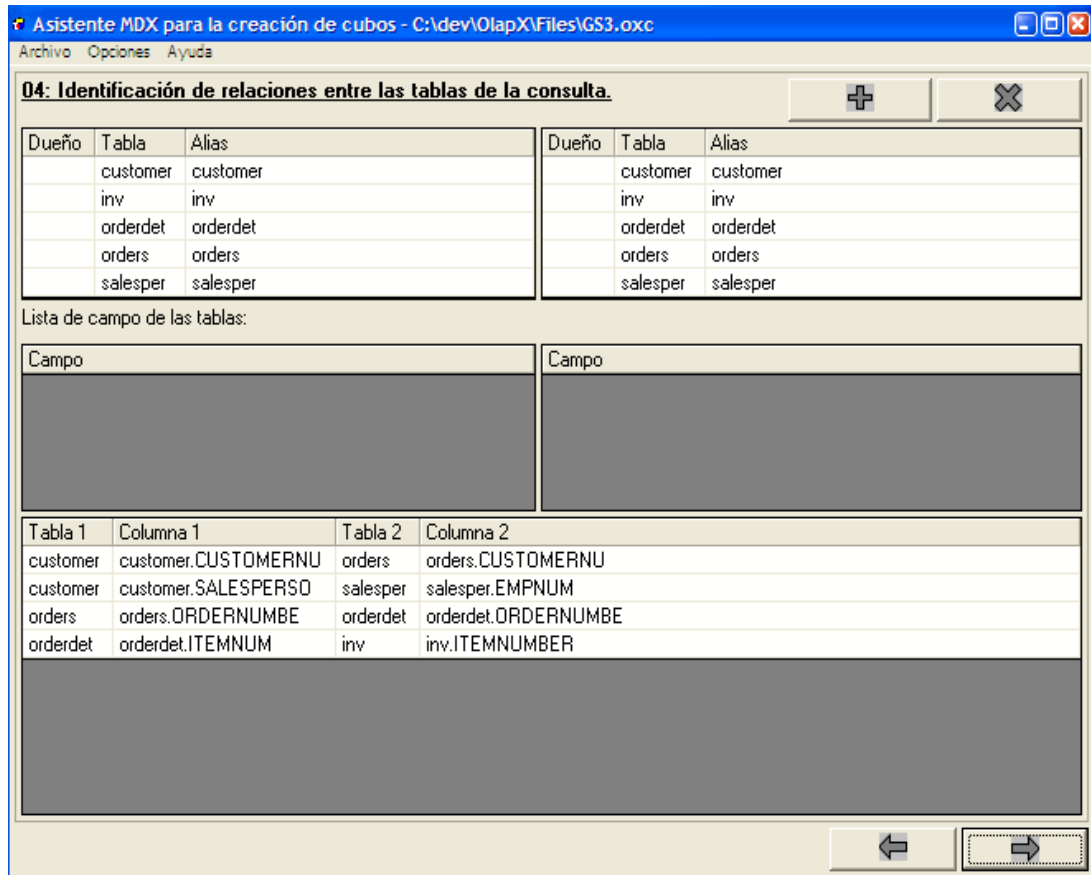
### **Identificación de relaciones entre las tablas de la consulta**

Esta ventana se divide en cinco secciones. En las primeras dos se muestran las tablas que se escogieron para la consulta. Al escoger una tabla de cada una de estas secciones, se muestran los campos de la tabla en las siguientes dos secciones. Aquí se debe escoger un campo de cada una de las tablas y oprimir el botón **Agregar** para crear la relación. Para quitar una relación de la lista se oprime el botón **Quitar**.

Se debe tener cuidado de escoger todas las relaciones entre todas las tablas para que así no se cree ningún producto cartesiano. Esto no quiere decir que todas las tablas tengan relación con todas las tablas, pero sí que debe haber un camino para relacionar cualquier tabla con las demás tablas. Por ejemplo:

Cliente -> Ordenes -> Detalle de ordenes -> Inventarios -> Proveedor

Con estas relaciones se ve qué mercancía de qué proveedores compran los clientes.



### Creación de preguntas para el usuario.

Esta pantalla se utiliza para definir preguntas al usuario, de tal forma que cada vez que se va a crear un cubo, se puede utilizar un criterio de consulta distinto cada vez. Por ejemplo, puede estar interesado solo en los datos del último mes o sólo de una sección de los datos.

### Creación de criterios de selección de registros

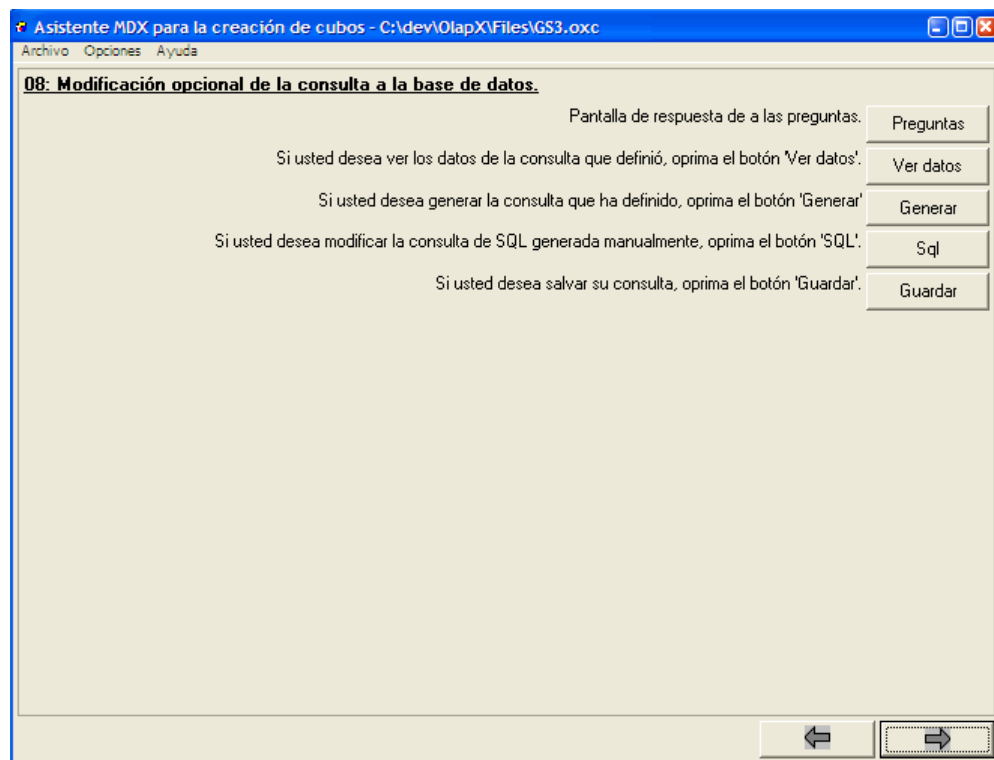
No siempre se desea traer todos los datos de las tablas que se escogieron para crear el cubo. Algunas veces se utiliza solo un subconjunto de ellos, por ejemplo los datos de una sola región de ventas, o los datos de cierta fecha o periodo en adelante. Esta sección permite escoger qué registros de la base de datos fuente se quiere utilizar para crear el cubo.

## Selección de campos para ordenar la consulta

En esta sección se puede especificar cómo se van a ordenar los datos de la consulta. Aunque no es necesario ordenar los datos para efectos de creación del cubo.

## Modificación opcional de la consulta a la base de datos

En este punto ya se ha terminado de definir la consulta a la base de datos que se va a utilizar para traer los datos que van a crear el cubo. Se observan diferentes opciones:



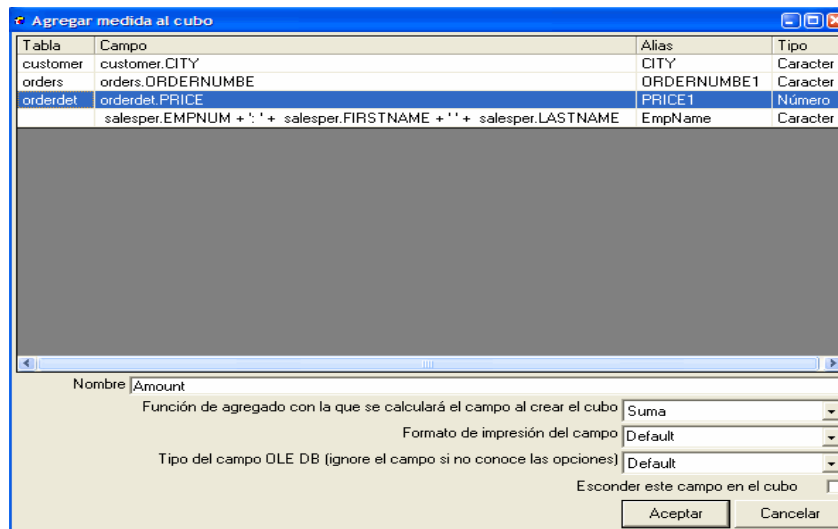
## Creación de la dimensión de medidas del cubo

En esta pantalla se va a definir la dimensión de medidas del cubo. Las medidas del cubo son campos de tipo numérico que tienen los valores que se van a medir, en otras palabras campos que se pueden sumar (hechos). Por ejemplo, la suma de ventas, los costos, importes, etc.

Al entrar a esta sección, automáticamente se establecen todos los campos numéricos como mediciones.

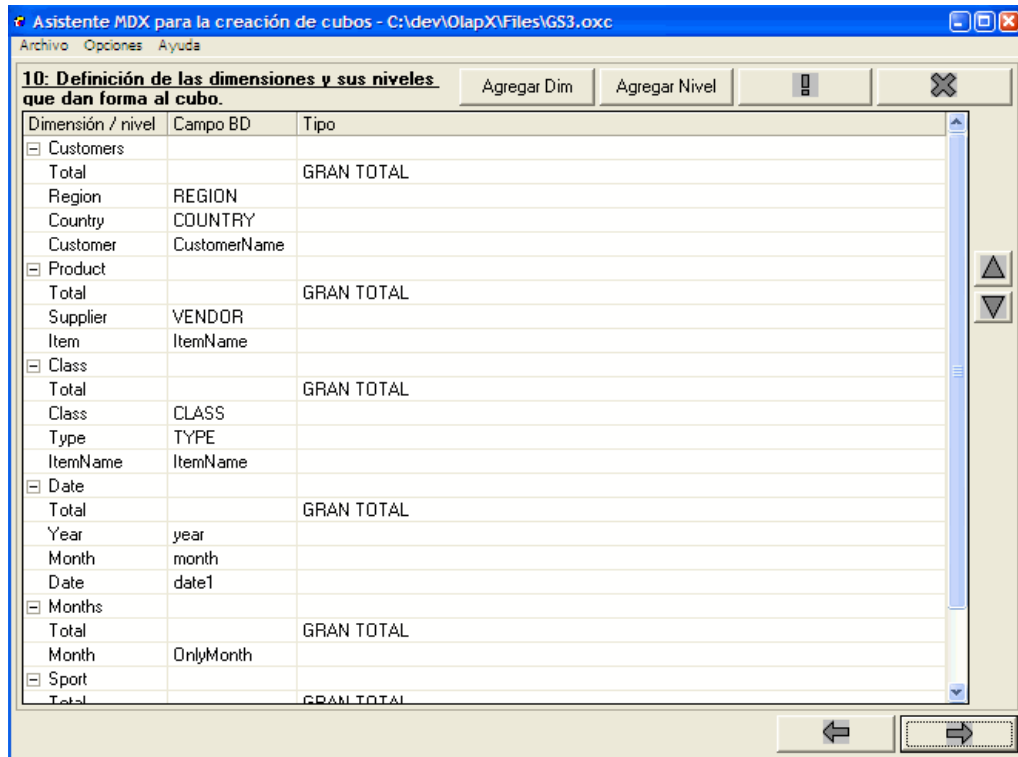


**Agregar una medida al cubo**



## Definición de las dimensiones y sus medidas que dan forma al cubo

Se definen las distintas dimensiones del cubo. Los cubos pueden tener varias dimensiones. Cada dimensión puede tener niveles, que vienen a dar forma a la dimensión. El nivel se utiliza para hacer drill-down sobre una dimensión y ver más detalle de datos, pasando de un nivel superior a un nivel inferior dentro de la dimensión.

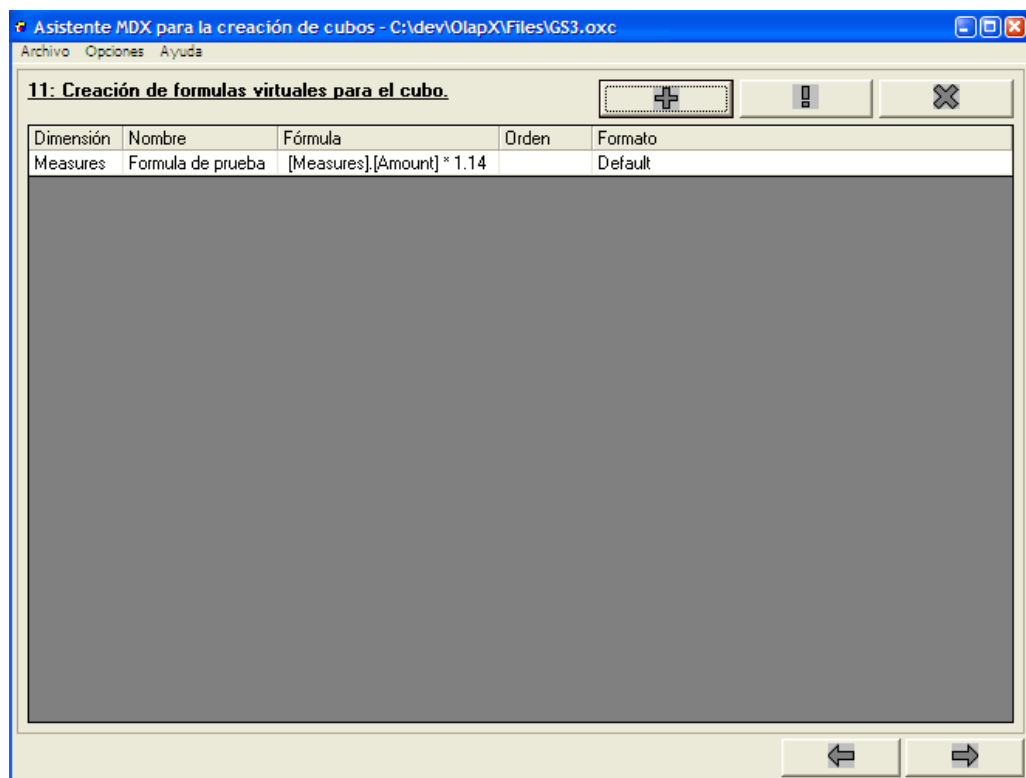


- **dimensión / Nivel:** Es el nombre de la dimensión o el nivel que aparece en el cubo.
- **Campo:** Especifica el campo de la base de datos, cuyos valores forman el nivel. Si la dimensión es una dimensión de tipo **tiempo**, entonces especifica el campo tipo fecha del cual se crea la dimensión.
- **Tipo:** Es el tipo de dimensión o nivel. Las dimensiones pueden ser de **TIEMPO** o normales, mientras que los niveles pueden ser una parte del tiempo en las dimensiones de tiempo o de tipo **GRAN TOTAL** que especifican el gran total de la dimensión.

## Creación de fórmulas virtuales para el cubo

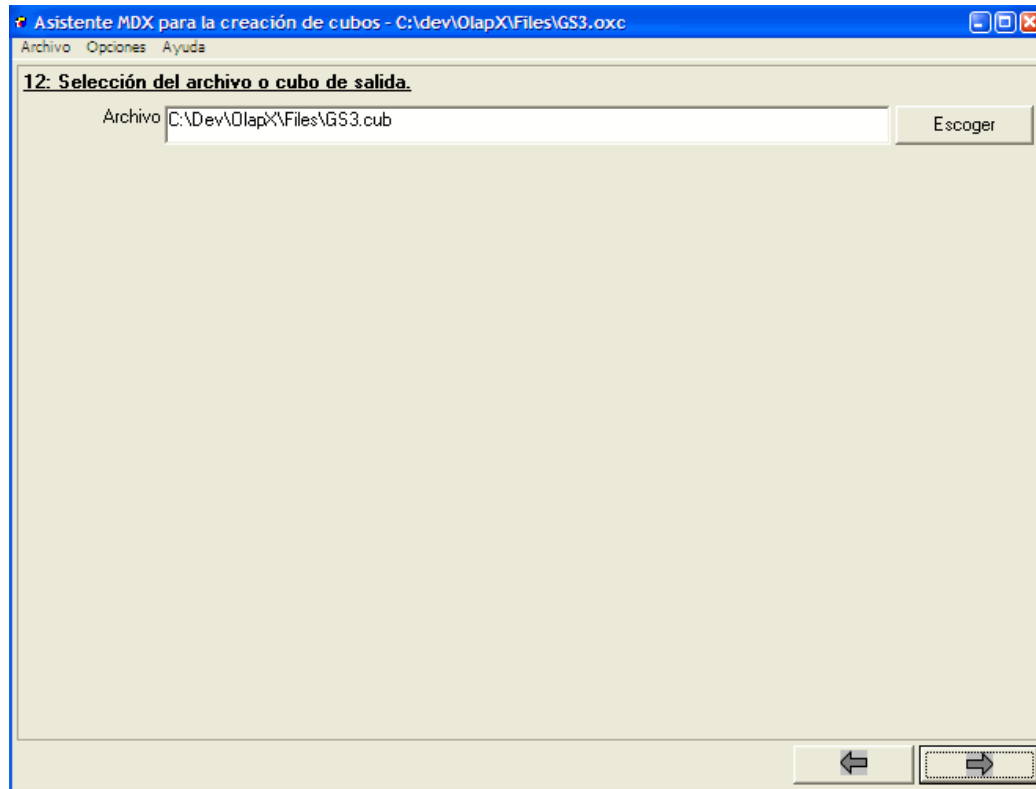
En esta sección se pueden crear campos virtuales cuya definición va a formar parte del cubo y desde el punto de vista del usuario del cubo, estos campos se van a ver como cualquier otro campo, sin saber que cada vez que se quiera utilizar el campo, se va a calcular la fórmula. En esta pantalla existen las siguientes opciones:

- **Dimensión:** dimensión a la cual pertenece este campo virtual.
- **Nombre:** nombre de este campo en el cubo.
- **Fórmula:** formula MDX que se va a aplicar al utilizar este campo en el cubo.
- **Orden:** es el orden en que se ejecutan todos los campos calculados. De esta forma se puede especificar si un campo depende de otro.
- **Formato:** es el formato con el que se pinta el campo calculado en el cubo.



## Selección del archivo o cubo de salida

Se selecciona el cubo que se va a crear:



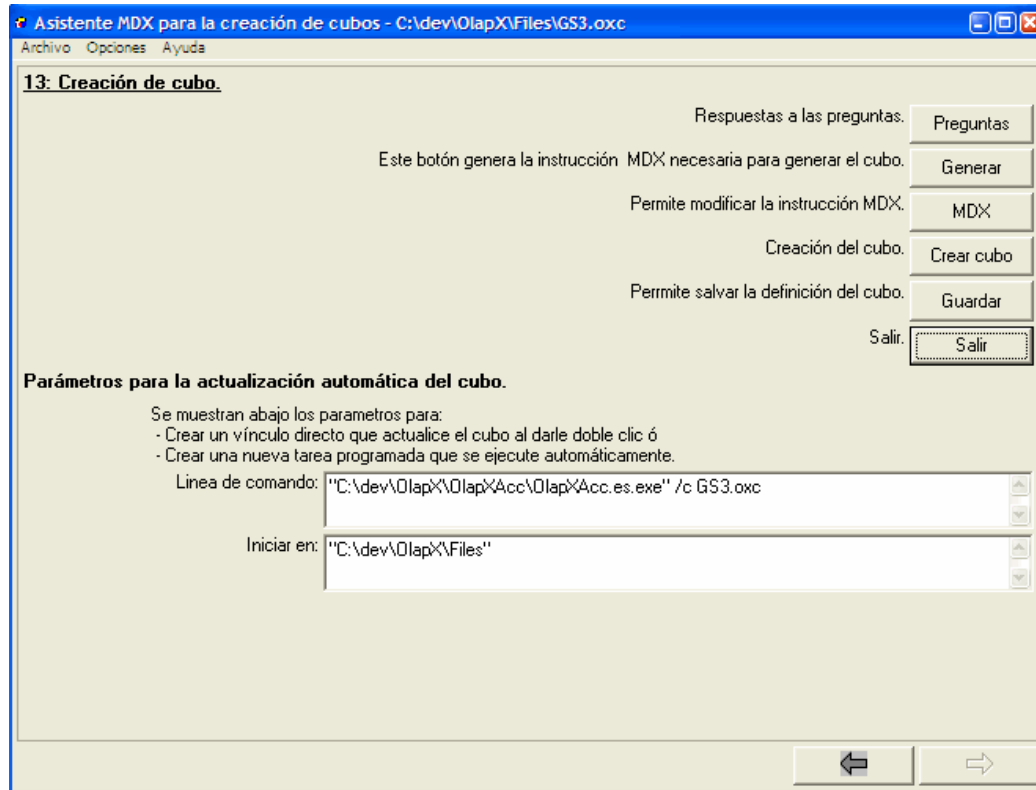
Las opciones avanzadas permiten:

Crear varios cubos dependiendo del valor de los campos de la base de datos. Por ejemplo, si se desea dar a cada gerente regional de su compañía un cubo que contenga solamente la información de su región. Entonces usted puede hacer un corte por el campo que trae la región y crear un cubo para cada región. Para utilizar esta opción se hace lo siguiente:

- **¿Desea crear un campo de corte para la división de los cubos según este valor?:** Se debe marcar esta opción si desea cortar el cubo en mini cubos.
- **Campo:** este es el campo de la base de datos que se va a utilizar para usar sus valores como nombre de cubos.
- **Escoger:** se escoge el campo de corte.
- Deseo utilizar un directorio distinto para cada cubo que se cree.



Por último se da clic en crear cubo para obtener el archivo que contenga los datos del cubo.



### Ventajas

- Posee muchas características para manipulación de los datos.

### Desventajas

- La persona que maneje este tipo de software debe estar muy bien capacitada para llevar a cabo una buena construcción de un cubo de datos, ya que las interfaces, procedimientos, funciones y demás no son tan intuitivos como en otras aplicaciones.
- Se requiere de muchos pasos y procedimientos para crear un cubo (13 pasos).
- No utiliza forma gráfica para la presentación de la información de las bases de datos (data warehouse).

### **3. METODOLOGIA, LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE STARCUBE**

#### **3.1 METODOLOGÍA DE DESARROLLO**

El Proceso Unificado de Rational o RUP (Rational Unified Process). Los orígenes de RUP se remontan al modelo espiral, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado.

Sus principales características son:

- ✓ Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- ✓ Pretende implementar las mejores prácticas en Ingeniería de Software
  - Desarrollo iterativo
  - Administración de requisitos
  - Uso de arquitectura basada en componentes
  - Control de cambios
  - Modelado visual del software
  - Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- ✓ inicio: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos
- ✓ elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos
- ✓ construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

- ✓ transición: se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.
- ✓ fases del RUP:
  - Establece oportunidad y alcance
  - Identifica las entidades externas o actores con las que se trata
  - Identifica los casos de uso

### 3.2 METODOLOGÍA DE ANÁLISIS Y DISEÑO

En STARCUBE se utiliza la metodología de análisis y diseño orientada a objetos, la cuál es extractada de las principales metodologías existentes, en particular:

- Object Oriented Design del autor Grady Booch [10], en la cual se desarrollan las tareas de análisis de requerimientos, análisis de dominio y diseño.
- Objectory del autor Ivar Jacobson [11], en la cual se desarrollan las tareas de análisis de requerimientos, análisis de robustez, diseño, implementación y pruebas.
- Object Modeling Technique del autor James Rumbaugh [10], en la cual se desarrollan las tareas de análisis, diseño del sistema, diseño de objetos e implementación.

Estas tres metodologías se fusionaron a finales de 1997 en una sola, basada en la notación UML.

#### **Etapas y actividades en el desarrollo orientado a objetos basado en UML**

Las etapas comprendidas en esta metodología son las siguientes:

- Análisis de requerimientos
- Diseño del sistema
- Diseño detallado
- Implementación y pruebas

Análisis de requerimientos: En esta etapa se logra claridad sobre lo que desea el usuario y la forma en la cual se le va a presentar la solución que está buscando.

- Actividades técnicas:
  - Identificar casos de uso del sistema.
  - Dar detalle a los casos de uso descritos.
  - Definir, si es aplicable, una interfaz inicial del sistema.
  - Desarrollar el modelo del mundo.
  - Validar los modelos.
  - Documentos entregables:
  - Casos de uso iniciales.

- Borradores de interfaz.
- Modelo del mundo inicial.

Diseño del Sistema: En esta etapa se define una subdivisión en aplicaciones del sistema (si es lo suficientemente grande) y la forma de comunicación con los sistemas ya existentes con los cuales debe interactuar.

- Actividades técnicas:
  - Identificar la arquitectura del sistema.
- Documentos entregables:
  - Diagramas de ejecución versión inicial.

Diseño detallado: En esta etapa se adecúa el análisis a las características específicas del ambiente de implementación y se completan las distintas aplicaciones del sistema con los modelos de control, interfaz o comunicaciones, según sea el caso.

- Actividades técnicas:
  - Agregar detalles de implementación al modelo del mundo.
  - Desarrollar el modelo de interfaz.
  - Desarrollar los modelos de control, persistencia y comunicaciones.
- Documentos entregables:
  - Diagramas de clases y paquetes, con el detalle de la implementación.
  - Diagramas de interacción con el detalle de las operaciones más importantes del sistema.

Implementación y pruebas: Se desarrolla el código de una manera certificada.

- Actividades técnicas:
  - Definir estándares de programación.
  - Codificación y pruebas unitarias.
  - Pruebas de módulos y del sistema.
- Documentos entregables:
  - Código fuente.
  - Soporte de pruebas unitarias.
  - Documentación del código.

### 3.3 LENGUAJE UNIFICADO DE MODELADO

El lenguaje unificado de modelado es una consolidación de muchas de las notaciones y conceptos orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares.

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

En UML se cuenta con 9 diagramas que sirven modelar los diferentes tipos de sistemas:

- Diagramas de casos de uso. Son utilizados para modelar los procesos 'business'.
- Diagramas de secuencia. Son utilizados para modelar el paso de mensajes entre objetos.
- Diagramas de colaboración. Son utilizados para modelar interacciones entre objetos.
- Diagramas de estado. Son utilizados para modelar el comportamiento de los objetos en el sistema.
- Diagramas de actividad. Son utilizados para modelar el comportamiento de los casos de uso, objetos u operaciones.
- Diagramas de clases. Son utilizados para modelar la estructura estática de las clases en el sistema.
- Diagramas de objetos. Son utilizados para modelar la estructura estática de los objetos en el sistema.
- Diagramas de componentes. Son utilizados para modelar componentes.
- Diagramas de implementación para modelar la distribución del sistema.

**Diagramas de clases:** El diagrama de clases es un modelo estático del sistema en el que se representan clases, interfaces, colaboraciones y relaciones.

Los diagramas de clases se utilizan para modelar el vocabulario de un sistema, las colaboraciones y esquemas lógicos de bases de datos.

Gráficamente una clase se representa mediante un rectángulo que contiene el nombre de la clase. El nombre de la clase es una cadena de texto que la identifica de manera única e inequívoca, esta cadena debe comenzar con letra minúscula y si está formado por varias palabras se escribirán sin espacio y cada una comenzará con letra mayúscula.

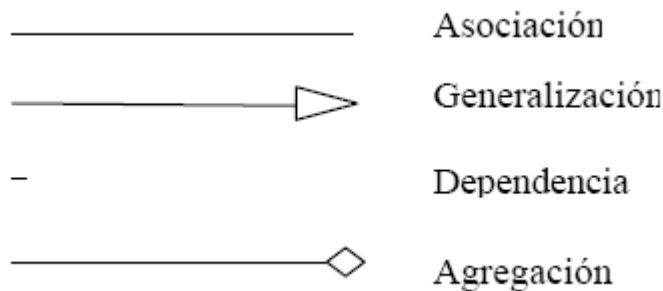
Cuando en el rectángulo solo aparece el nombre de la clase se dice que ésta tiene un nombre simple, cuando aparece el nombre de la clase precedido por el nombre del paquete, se denomina nombre de camino. Las clases abstractas se representan escribiendo el nombre en cursiva.

- **Atributos.** Una clase puede tener o no tener atributos, éstos pueden aparecer o no aparecer en el diagrama de clases. Si se incluyen en el

diagrama, se listan en un rectángulo bajo el nombre de la clase. Los atributos se escriben en letras minúsculas, incluso la primera letra, pero si están formados por más de una palabra a partir de la segunda tendrán la primera letra en mayúscula.

- **Operaciones.** Las operaciones representan acciones que desarrollan los objetos, por lo general se escriben como una expresión verbal. Es opcional incluirlos en el diagrama de clases, si se desea especificarlos, se escribirán en un rectángulo debajo de los atributos. El nombre de las operaciones se escribirá en minúscula, la primera letra de cada palabra será mayúscula, excepto para la primera. Las operaciones pueden especificar el tipo de retorno, si es que lo tienen, y los parámetros con sus tipos.
- **Responsabilidades.** Son un contrato u obligación de la clase, indican lo que ha de hacer cada objeto. Las responsabilidades se cumplen a través de las operaciones, las cuales pueden apoyarse en los atributos y los estados del objeto. Una clase debe tener como mínimo una responsabilidad, pueden ser más de una, pero es preferible que no sean muchas. Las responsabilidades se especifican de manera textual en un compartimiento debajo de las operaciones.

Las relaciones modelan conexiones físicas, semánticas o lógicas entre elementos de un diagrama. La notación para representarlas es la siguiente:



- **Asociación.** Es una relación estructural que indica que los objetos de una clase están conectados con los objetos de otra.
- **Generalización.** Es una relación que asocia una clase general con una clase especializada que posee la estructura y operaciones de la primera y que podría sustituirla.
- **Dependencia.** Es una relación de uso donde una clase depende de otra, es decir, que un cambio en la clase independiente tendrá efecto en la clase dependiente. Una relación de dependencia se presenta cuando una clase utiliza objetos de otra como argumentos de sus operaciones.

- **Agregación.** Es una especialización de la asociación entre dos clases que indica que los objetos de una clase forman parte de los objetos de la otra, es decir que estructuralmente un objeto contiene a otro.

**Diagrama de paquetes:** El Diagrama de paquetes, muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones o paquetes. Un paquete es un elemento de propósito general que se utiliza para agrupar elementos. Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes se dibujan como rectángulos, las dependencias se muestran como flechas con líneas discontinuas. El operador "::" permite designar una clase definida en un contexto distinto del actual.

**Diagrama de casos de uso:** Un caso de uso especifica el comportamiento del sistema o parte de él en relación con la intervención de un usuario, define un conjunto de acciones que se realizan para generar un resultado observable e importante para un actor.

Un diagrama de casos de uso muestra los casos de uso de un sistema o parte de él, los actores y las relaciones que se establecen entre ellos. Los límites del sistema se indican mediante un cuadro o un rectángulo, dentro de éste se coloca los casos de uso y fuera, los actores. Los casos de uso se representan mediante elipses y los actores mediante figuras humanas.

Un actor representa un rol desempeñado por algún usuario, dispositivo o sistema externo que interactúa con el sistema en estudio. Los casos de uso ayudan a identificar y modelar los requisitos funcionales del sistema por este motivo son ampliamente utilizados en las primeras fases del proceso de desarrollo de software. Los casos de uso pueden organizarse especificando relaciones entre ellos como: especializaciones, inclusiones y extensiones.

La especialización o relación de generalización en los casos de uso tiene el mismo significado y funcionalidad que en las clases. Dado un caso de uso base, puede tenerse otro especializado (hijo) que hereda el comportamiento del primero. El caso de uso especializado puede añadir funciones o redefinir las que ha heredado.

La representación de la generalización entre casos de uso se hace de igual manera que en las clases, mediante una línea continua terminada en punta de flecha vacía que se dirige desde el caso de uso especializado hasta el caso de uso general.

Las relaciones de inclusión se utilizan para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte. La inclusión consiste en delegar funciones.

Se identifica responsabilidades del sistema que figuren en más de un caso de uso,

se elabora un caso de uso para dichas funciones y luego se incluye en otros casos de uso que requieran desarrollar dichas funciones. Para indicar una relación de inclusión entre casos de uso se utiliza una dependencia con el estereotipo include.

Una relación de extensión indica que el comportamiento de un caso de uso base puede extenderse con el comportamiento de otro caso de uso que extiende al primero. Este tipo de relaciones se representan como una dependencia estereotipada con la palabra extend.

**Diagrama de secuencia:** Es un diagrama para mostrar una interacción entre un conjunto de objetos indicando el orden en que se envían y reciben los mensajes. Para elaborar el diagrama de secuencia se colocan en la parte superior, siguiendo el eje X, los objetos que participan en la interacción, teniendo en cuenta de colocar a la izquierda el objeto que inicia la interacción y los objetos subordinados hacia la derecha. Debajo de cada objeto y siguiendo el eje Y, se traza la línea de vida y el foco de control.

- La línea de vida. Es una línea discontinua vertical, trazada debajo de un objeto e indica la existencia de éste en el tiempo. Algunos objetos existen durante toda la interacción, en cuyo caso el objeto se colocará en la parte superior del diagrama y la línea de vida se trazará de arriba hacia abajo hasta el fin del diagrama. Algunos otros objetos serán creados y destruidos durante la interacción, en este caso el objeto se colocará a la altura del mensaje que lo crea y su línea de vida se extenderá a partir de esa posición.

- El foco de control. Es un rectángulo delgado que se traza de forma vertical sobre la línea de vida para indicar el tiempo que el objeto está ejecutando una acción, ya sea que ejecute la acción con sus propios métodos o con métodos subordinados, es decir, invocando métodos en otros objetos. La parte superior del foco de control indica el comienzo de la acción y debe estar a la altura del primer mensaje, mientras que la parte inferior corresponde al fin de la acción y puede marcarse con un mensaje de retorno.

**Diagramas de colaboración:** Representan la organización de los objetos que participan en una interacción. Estos diagramas incluyen los mensajes que se envían entre los objetos, pero no se enfocan en el orden temporal de éstos. Un diagrama de colaboración se construye con los objetos que participan en la interacción y los enlaces que se dan entre ellos. Los mensajes que se envían entre objetos se colocan como adornos de los enlaces, utilizando un número para representar el orden en que se generan los mensajes.

Los enlaces entre los objetos corresponden a las relaciones entre clases en el diagrama de clases y suele ser suficiente para que los objetos puedan enviarse mensajes entre sí. Se dice que el enlace es el camino por donde se envían los mensajes.



### 3.4 LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación para la implementación de la herramienta es Java™, el cuál es en sí una plataforma independiente tanto de hardware como de software gracias al soporte de máquina virtual.

Java™ es un lenguaje de programación orientado a objetos que surgió en 1991 cuando un grupo de ingenieros de Sun Microsystems trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Java™, como lenguaje de programación para computadores, se introdujo a finales de 1995. La clave fue la incorporación de un intérprete Java™ en el programa Netscape™ Navigator, versión 2.0, produciendo una verdadera revolución en Internet. Java™ 1.1 apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. Al programar en Java™ no se parte de cero.

Cualquier aplicación que se desarrolle se apoya en un gran número de clases preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el API o Application Programming Interface de Java™).

Java™ incorpora muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads o hilos, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso es un lenguaje ideal para aprender la informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo.

El principal objetivo del lenguaje Java™ es llegar a ser el “nexo universal” que conecte a los usuarios con la información, esté ésta situada en el ordenador local, en un servidor de Web, en una base de datos o en cualquier otro lugar. Java™ es un lenguaje muy completo que se está convirtiendo en un macro-lenguaje: Java™ 1.0 tenía 12 packages (paquetes); Java™ 1.1 tenía 23 y Java™ 1.2 tiene 59. La compañía Sun describe el lenguaje Java™ como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”.

#### 3.4.1 Principales características de Java™.

Lenguaje orientado a objetos. Es una característica en la que el software se

desarrolla de forma que los distintos tipos de datos que use están unidos a sus operaciones, de esta manera las clases, datos y el código se combinan en entidades llamadas objetos.

Una clase es una agrupación de datos (variables, campos o elementos declarados de la clase) y de funciones (métodos) que operan sobre esos datos. A estos datos y funciones pertenecientes a una clase se les denomina variables y métodos o funciones miembro.

La programación orientada a objetos se basa en la programación de clases. Un programa se construye a partir de un conjunto de clases. Esto hace posible que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. La herencia permite que se puedan definir nuevas clases basadas en clases existentes, lo cual facilita re-utilizar código previamente desarrollado.

Independencia de la plataforma. Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, es importante conseguir una herramienta independiente del tipo de procesador utilizado. El desarrollo de un código “neutro” que no depende del tipo CPU utilizada permitió lograr lo que luego se ha convertido en el principal lema del lenguaje: “Write Once, Run Everywhere”.

De esta manera se logra obtener un código capaz de ejecutarse en cualquier tipo de máquina, el cual una vez compilado no necesita de ninguna modificación por el hecho de cambiar de procesador o de ejecutarlo en otra máquina. La clave consiste en desarrollar un código “neutro” el cual está preparado para ser ejecutado sobre una “máquina hipotética o virtual”, denominada Java Virtual Machine (JVM).

Es esta JVM quien interpreta este código neutro convirtiéndolo a código particular de la CPU o chip utilizado, evitando así el tener que realizar un programa diferente para cada CPU o plataforma. La JVM es el intérprete de Java™. Ejecuta los “bytecodes” (archivos compilados con extensión \*.class) creados por el compilador de Java™ (javac.exe). Tiene numerosas opciones entre las que destaca la posibilidad de utilizar el denominado JIT (Just-In-Time Compiler) que interpreta el bytecode generado convirtiéndolo a instrucciones máquina del código nativo. El JIT puede mejorar entre 10 y 20 veces la velocidad de ejecución de un programa.

El recolector de basura: Uno de los aspectos más importantes en la programación orientada a objetos (OOP) es la forma en la cual son creados y eliminados los objetos. La eliminación de los objetos la realiza el denominado garbage collector, quien automáticamente libera o borra la memoria ocupada por un objeto cuando no existe ninguna referencia apuntando a ese objeto. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java™ (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos.

Cuando no quedan referencias a un objeto, el recolector de basura de Java™ borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas de memoria, que ocurre cuando el sistema operativo piensa que una zona de memoria está siendo usada por una aplicación cuando en realidad no es así. Concluyendo se puede decir que el recolector de basura de Java™ permite una fácil creación y eliminación de objetos de una manera rápida y segura.

### 3.5 ENTORNO DE DESARROLLO NETBEANS IDE 6.0

El entorno de desarrollo que se escogió para la implementación de la herramienta STARCUBE es NetBeans IDE 6.0, un software bajo licencia CDDL v1.0 creado para desarrolladores de software.

NetBeans es una plataforma para el desarrollo de aplicaciones profesionales de escritorio, empresariales, aplicaciones web y móviles, utilizando lenguajes como: Java, C/C++; el entorno de desarrollo corre sobre muchas plataformas: Windows, Linux, Mac OS X y Solaris.

El entorno de desarrollo integrado (IDE), posibilita editar programas, compilarlos, ejecutarlos, depurarlos y construir rápidamente la gráfica de una aplicación.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.

Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas

### 3.6 CONTROLADOR JDBC

JDBC es el acrónimo de *Java Database Connectivity*, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el lenguaje SQL del modelo de base de datos que se utilice.

La API JDBC consta de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. En otras palabras, con el API JDBC no es necesario escribir un programa para acceder a DB2, otro programa para acceder a Oracle, y otro programa para acceder a MySQL; con esta API, se puede crear un sólo programa en Java que sea capaz de enviar sentencias SQL a la base de datos apropiada.

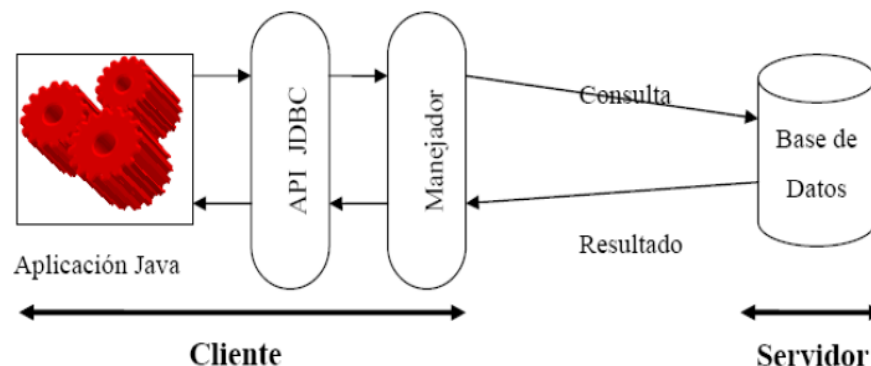
La aplicación de Java debe tener acceso a un controlador (*driver*) JDBC adecuado. Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real. De manera muy simple, al usar JDBC se pueden hacer tres cosas:

- Establecer una conexión a una fuente de datos
- Mandar consultas y sentencias a la fuente de datos.
- Procesar los resultados.

Los distribuidores de bases de datos suministran los controladores que implementan el API JDBC y que permiten acceder a sus propias implementaciones de bases de datos. De esta forma JDBC proporciona a los programadores de Java una interfaz de alto nivel y evita el tener que tratar con detalles de bajo nivel para acceder a bases de datos.

Para la implementación de las conexiones con la bases de datos PostgreSQL se utilizará la clasificación JDBC que utiliza el Manejador de Protocolo Nativo (tipo 4), este traduce directamente las llamadas al API JDBC al protocolo nativo de la base de datos. Es el manejador que tiene mejor rendimiento, pero está más ligado a la base de datos que se emplee. En la figura 2.8 se observa la arquitectura Manejador de Protocolo Nativo tipo 4.

Figura 2.8. Arquitectura manejador de protocolo nativo tipo 4.



### 3.7 JFREECHART

JFreeChart es una biblioteca gráfica de Java 100% libre, la cual facilita a los desarrolladores mostrar gráficos de calidad profesional en sus aplicaciones.

El amplio conjunto de características de JFreeChart incluye:

- Una consistente y bien documentada API, dando soporte para una amplia gama de tipos de gráficos.
- Un diseño flexible que es fácil de extender, abarcando aplicaciones tanto del lado del servidor como del lado del cliente.
- Soporte para muchos tipos de salida, incluyendo los componentes Swing, archivos de imagen (incluyendo PNG y JPEG), gráficos vectoriales y formatos de archivo (incluyendo PDF, EPS y SVG).
- JFreeChart es "código abierto" o, más concretamente, software libre. Se distribuye bajo los términos de la Licencia LGPL (GNU Lesser General Public Licence), que permite su uso en aplicaciones propietarias.

El proyecto JFreeChart fue fundado en febrero del año 2000 por David Gilbert. Hoy en día, JFreeChart es utilizada por aproximadamente 40.000 a 50.000 desarrolladores. El proyecto sigue siendo administrado por Gilbert, con contribuciones de una amplia comunidad de desarrolladores.

### 3.8 OLAP4J

OLAP4j es una API de Java para la creación de aplicaciones OLAP. En esencia, OLAP4j es para datos multidimensionales como lo es JDBC para datos relacionales. OLAP4j tiene un modelo de programación similar a JDBC, comparte algunas de sus principales clases, y tiene muchas de las mismas ventajas. Se puede escribir una aplicación OLAP en Java para un servidor (por ejemplo Mondrian) y cambiar fácilmente a otro (por ejemplo Microsoft Analysis Services, accesible a través de XML for Analysis).

Sin embargo, la creación de un estándar de Java API para OLAP es una cuestión polémica [15].

## **Inicios de los estándares para API's OLAP**

La historia está llena de intentos de crear un estándar de una API para OLAP. En primer lugar está el consejo de OLAP MDAPI (en dos versiones), luego emergió la API JOLAP de Sun Java Community Process. Pero todos estos intentos fallaron, al parecer porque en algún punto durante las etapas de la comisión, todos los vendedores de servidores OLAP perdieron el interés en la implementación de un estándar.

Los estándares eran grandes y complejos, y ningún proveedor de interfaces de usuario dio un paso adelante con una IU que trabajara con múltiples back-ends.

Mientras tanto, Microsoft introdujo OLE DB para OLAP (que sólo funciona entre clientes y servidores de Windows) y, a continuación, XML / A (XML para el análisis, una API de servicios web).

Estos estándares tuvieron más éxito, por varias razones. En primer lugar los estándares (OLE DB para OLAP, en particular) fueron impulsados principalmente por un sólo vendedor, no eran un compromiso de solución tratando de abarcar la funcionalidad de varios productos.

En segundo lugar, había una implementación de referencia, y Microsoft vio que había suficientes clientes OLAP para hacer de éstos estándares foros viables de competencia e innovación. En tercer lugar, estaba el lenguaje de consulta MDX (un lenguaje de consulta es más fácil de explicar que una API) que deja sin resolver el problema de cómo construir consultas (queries) para responder preguntas de negocio, pero los desarrolladores podrían resolver ese problema integrando algún cliente OLAP.

La comunidad Open Source ha desarrollado pruebas para OLAP. En primer lugar, existió Mondrian, un servidor OLAP de código abierto y luego JPivot, un cliente que primero se comunicó con Mondrian, luego también con XML/A. Luego había más clientes OLAP y aplicaciones que querían utilizar un cliente en particular, pero a la vez querían comunicarse con una variedad de servidores, y empresas que utilizaban un servidor OLAP que quería soportar varios tipos de clientes.

Se puso de manifiesto que las herramientas OLAP de código abierto necesitaban un estándar, y que ese estándar probablemente sería adecuado para otras herramientas OLAP basadas en Java [15].

## **Descripción de OLAP4J**

Una aplicación OLAP interactúa con un servidor OLAP a través de sentencias MDX pertenecientes a las conexiones. Las sentencias se definen en términos de metadatos y son validadas de acuerdo con un tipo de sistema, y algunas

aplicaciones son construidas en un nivel superior, manipulando parse-trees (árboles sintácticos) de MDX, y definiendo consultas complejas en términos que un usuario de negocios pueda entender. La API OLAP4j proporciona todas estas facilidades.

En el nivel más bajo, OLAP4j tiene un framework (espacio de trabajo) para el registro de los drivers, y el manejo del ciclo de vida de las conexiones y las sentencias. OLAP4j proporciona este soporte mediante la ampliación del framework de JDBC.

Una decisión clave en el diseño de una API de OLAP es si se debe incluir un lenguaje de consulta. Históricamente ha sido una polémica. Los estándares anteriores se dividían en dos campos: MDAPI y JOLAP tenían una API para la construcción de consultas, mientras que OLE DB para OLAP y XML/A tenía el lenguaje de consultas MDX. El lenguaje de consulta SQL es un componente esencial de las API's de bases de datos relacionales como son ODBC y JDBC, y hay un sentido similar al basar la API OLAP en un lenguaje de consulta como MDX.

Pero las aplicaciones OLAP también necesitan crear y transformar las preguntas cuando el usuario final explora los datos. Por lo tanto, OLAP4j abarca ambos enfoques: se puede crear una consulta de MDX analizando una sentencia y se puede construir una consulta MDX manipulando un parse-tree, además de que una biblioteca de parsing de MDX permite convertir fácilmente una cadena de MDX a un parse-tree y desde un parse-tree.

Los Metadatos son el corazón de OLAP4j. Se puede navegar por los cubos, las dimensiones, las jerarquías, los miembros de un esquema OLAP, y los parse-tree de MDX y los resultados de la consulta están vinculados a los mismos objetos de metadatos. También existe un tipo de sistema para la descripción de las expresiones.

OLAP4j permite programar un cliente OLAP sin partir desde cero. Además del parser MDX, y las operaciones sobre el parse-tree de MDX, hay un modelo de consulta de mayor nivel, que incluye operaciones de transformación de las consultas (también llamadas 'navigaciones'), además de proporcionar facilidades para la disposición de resultados multidimensionales como tablas HTML [15]. En la figura 2.9 se visualiza la arquitectura de la API OLAP4J.

### **Beneficios de una API de Java estándar para OLAP**

Una vez que el estándar OLAP4j está en su lugar, se puede esperar los beneficios conocidos que surgirán de un estándar abierto: una mayor variedad de herramientas, mejores herramientas, y más competencia entre servidores OLAP en

cuanto a precios y características. Estos beneficios continúan porque si un desarrollador de herramientas OLAP puede alcanzar una audiencia más amplia, hay mayores incentivos para construir nuevas herramientas.

Eventualmente habrá proveedores OLAP4j para la mayoría de los servidores OLAP. Los vendedores de servidores inicialmente tendrán pocos incentivos para adoptar un estándar que introducirá competencia en su mercado, pero con el tiempo, la riqueza de las herramientas les obligará a crear uno, o, más probablemente, atraerá a terceros o a los esfuerzos open-source para construir proveedores para sus servidores [15].

## **OLAP4J y MDX**

Todos los drivers de OLAP4j deben implementar las características fundamentales del lenguaje MDX, pero los niveles de cumplimiento de OLAP4j no implican un grado particular de soporte para el lenguaje MDX.

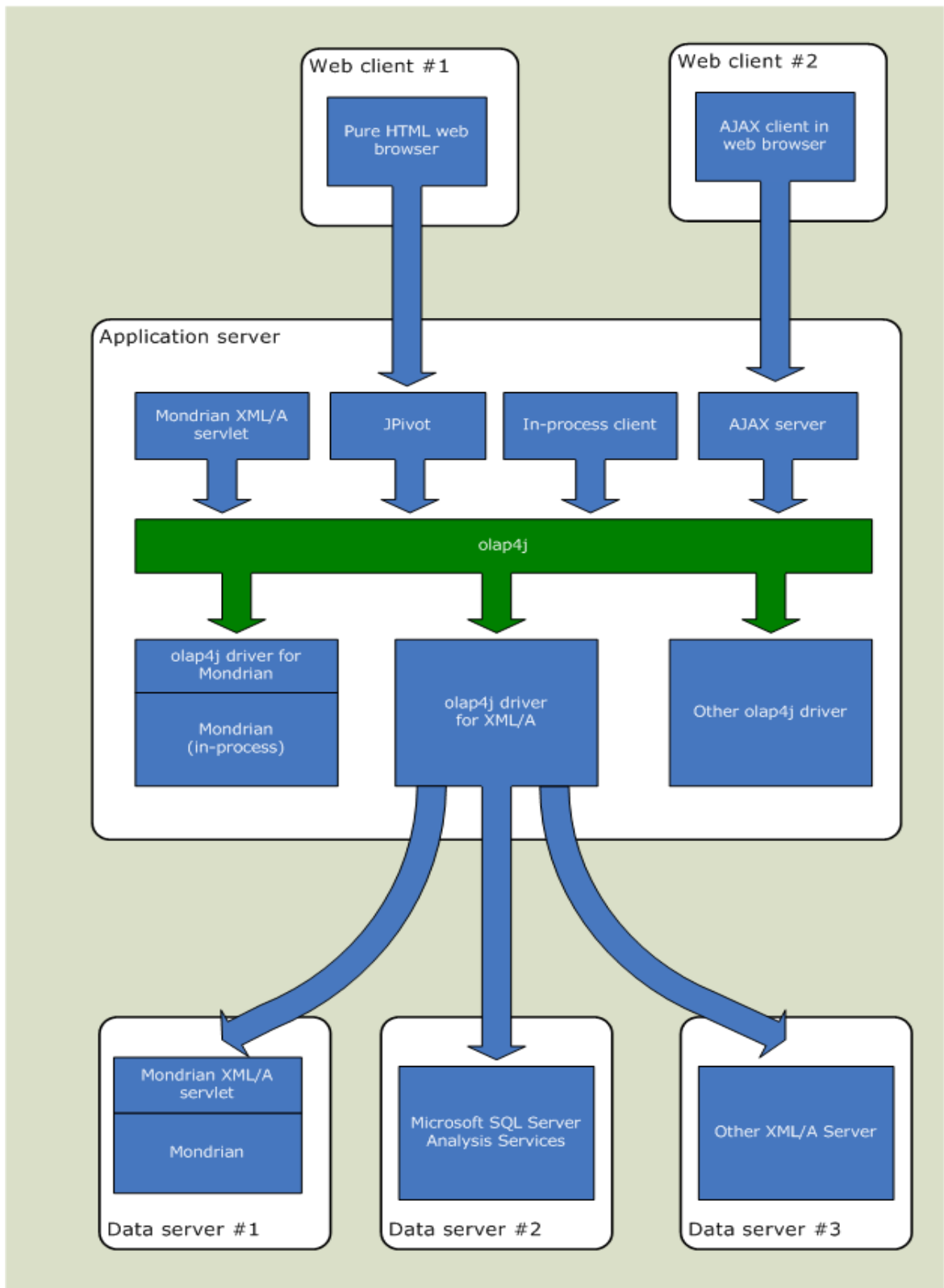
Las características fundamentales de MDX son:

- Consultas de la forma "SELECT ... FROM ... WHERE".
- El operador constructor de conjunto { ... } .
- Establecer las funciones CrossJoin(<Set>, <Set>), Filter(<Set>, <Condition>), Order(<Set>, <Expression>), Hierarchize(<Set>).
- Operadores de navegación: <Member>.Children, <Level>.Members <Hierarchy>.Members, <Member>.Parent, Level, Hierarchy, Dimension.
- Funciones de agregación: Aggregate
- Aritmética básica y operadores lógicos.

Por supuesto, un proveedor puede aplicar un gran subconjunto de MDX, y la mayoría lo hacen. Un proveedor puede describir el nivel de cumplimiento de su MDX, describiendo características adicionales que éste puede soportar (por ejemplo, las cláusulas WITH MEMBER, WITH SET, NON EMPTY, y HAVING en: consultas, cubos virtuales, miembros calculados y conjuntos definidos para los cubos) y funciones adicionales y operadores implementados [15].



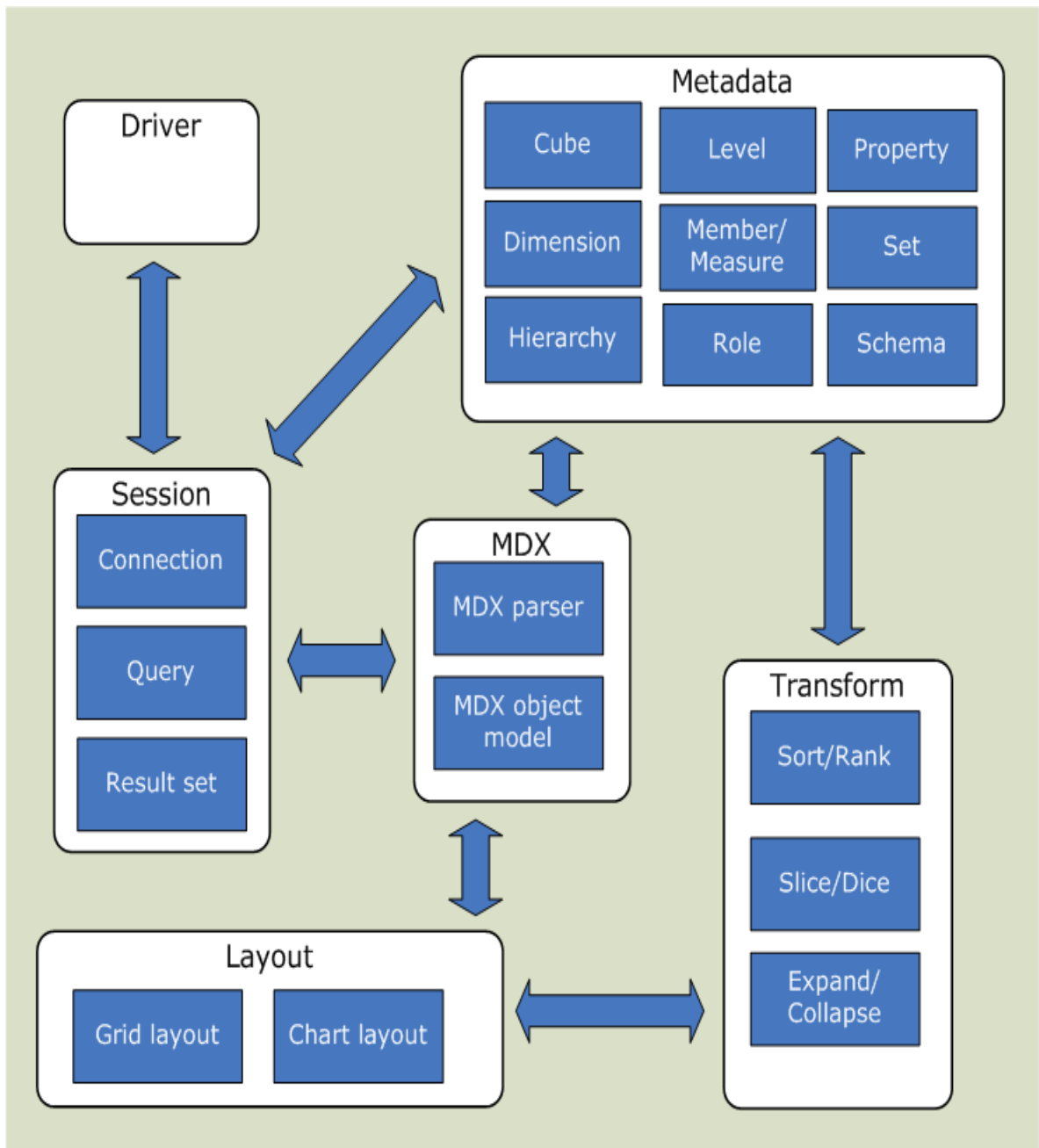
Figura 2.9. Arquitectura de la API OLAP4J



## Componentes de la API

En la figura 2.10, se describe la API OLAP4j con más detalle, discriminada en una serie de áreas funcionales [15].

Figura 2.10. Componentes de la API OLAP4J



### 3.9 MYDOGGY

MyDoggy Java es un framework (marco de trabajo) de acoplamiento para ser utilizado en aplicaciones Swing de plataforma cruzada. Es un espacio para la gestión de ventanas secundarias dentro de una ventana principal.

MyDoggy permite mover, cambiar el tamaño o extraer las ventanas secundarias. Además, presta apoyo para la gestión de contenidos de la ventana principal y apoya el concepto de "prospectiva" utilizando grupos.

#### Módulos principales

MyDoggy se separa en dos módulos principales:

- **MyDoggy-API:** Contiene la interfaz de aplicación de MyDoggy para gestionar cada aspecto del framework.
- **MyDoggy-Plaf:** Contiene la aplicación por defecto de mydoggy-api.
  - **MyDoggy-res:** Contiene todos los recursos (iconos y mensajes) utilizados por mydoggy-plaf.

#### Otros módulos

También existen otros módulos que cubren aspectos secundarios de MyDoggy.

- **Ejemplos MyDoggy:** Contiene algunos ejemplos MyDoggy para ver en acción.

La **API-MyDoggy** contiene la interfaz de programación de la aplicación MyDoggy para gestionar todos los aspectos de un framework (espacio de trabajo). Las clases principales utilizadas para manipular MyDoggy son:

- **ToolWindowManager:** Esta interfaz es el principal punto de entrada para administrar MyDoggy. Usando esta interfaz el usuario puede registrar o no registrar grupos de "tool windows" (ventanas de herramientas). El usuario puede obtener el "content manager" (gestor de contenidos) y el tipo de plantillas.
- **ToolWindow:** Esta interfaz modela el concepto de una ventana secundaria que proporciona el acceso y/o soporte para funcionalidades específicas. Esta interfaz es el punto de entrada principal para modificar las propiedades de la "tool window". Además, hay métodos para hacer la herramienta disponible, visible y activa.

- **ToolWindowGroup:** Esta interfaz permite al usuario administrar un grupo de *tool window*. La idea es proporcionar el mismo mecanismo de las perspectivas de la IDE Eclipse para *toolwindows*. Este componente es utilizado típicamente para recuperar un grupo desde el gestor de *toolwindows* y también para añadir cualquier número de *toolwindows*. Después, se podrá mostrar u ocultar todas las herramientas que estén registrados en un grupo.
- **ContentManager:** Este administrador gestiona la ventana principal. Es posible añadir varios contenidos para ser mostrados en esta ventana. El usuario puede cambiar entre los contenidos mostrados. Los contenidos son agregados al *ContentManager* utilizando el método *addContent*.
- **Content:** Un “content” (contenido) es una “envoltura” de un componente adornado con algunas propiedades como un título, un icono, etc. La visualización de un *content* específico depende de la plataforma de aplicación. Una plataforma de aplicación puede utilizar un *JTabbedPane* o un *JDesktopPane* por ejemplo.
- **ContentManagerUI:** Un “ContentManagerUI” es una interfaz para modificar el comportamiento de la UI (User Interface) de un *content manager*. Por ejemplo, este se utiliza para modificar la forma en que un contenido es mostrado.
- **PersistenceDelegate:** Esta interfaz proporciona métodos útiles para guardar y cargar el “framework” (espacio de trabajo) de un *tool window manager*. Cuando se solicita guardar el *framework*, todos los ajustes de las *ToolWindows* con descripciones relativas se guardan. Para obtener una instancia del *PersistenceDelegate* hay que invocar el método *getPersistenceDelegate* de *ToolWindowManager* interfaz. Así que usted puede obtener una persistencia delegado específicamente a un determinado instrumento gestor de ventanas.

**MyDoggy-Plaf** por defecto contiene la API por defecto de mydoggy.

Aquí se puede encontrar información como:

- **MyDoggyToolWindowManager:** La implementación de la interfaz *ToolWindowManager*. Esta interfaz es el principal punto de entrada para administrar MyDoggy. Usando esta interfaz el usuario puede registrar o des registrar *toolwindows* o *groups*. El usuario puede obtener el gestor de contenidos y hacer plantillas descriptivas.
- **XmlPersistenceDelegate:** Implementación de la interfaz *PersistenceDelegate*. Interfaz que proporciona métodos útiles para guardar

y cargar el espacio de trabajo de las *toolwindow*. Cuando se solicita guardar el espacio de trabajo, se almacenan todos los ajustes de las *ToolWindows* con relativas descripciones. Para obtener una instancia del *PersistenceDelegate* hay que invocar el método *getPersistenceDelegate* de la interfaz *ToolWindowManager*. Por lo tanto se puede obtener un *PersistenceDelegate* específico de determinado *ToolWindowManager*.

- **MyDoggyTabbedContentManagerUI:** Implementación de la interfaz *TabbedContentManagerUI*. Representa una interfaz IU que usa un componente que permite al usuario cambiar entre un grupo de componentes, haciendo clic en una pestaña con un determinado título y/o icono (es decir, un *JTabbedPane*).
- **MyDoggyDesktopContentManagerUI:** Implementación de la interfaz *DesktopContentManagerUI*. Representa una interfaz IU que utiliza un contenedor para crear una interfaz de múltiples documentos o un escritorio virtual.
- **Transparencia:** En la plataforma de Windows, a partir de Windows 2000, se puede ver la transparencia para una ventana flotante con su contenido.
- **Resource Manager:** Interfaz que se utiliza para personalizar no sólo los iconos, los colores y la internacionalización, sino también la creación de componentes de IU, en relación con la personalización y manejo de transparencia [8].

### 3.10 BIBLIOTECA GRÁFICA SWING DE JAVA™

Swing es una biblioteca gráfica para Java™ que hace parte de las Java™ Foundation Classes que comprende un grupo de componente para ayudar a construir interfaces gráficos de usuario (GUIs).

**JComponent:** La mayoría de los componentes Swing están implementados como subclases de la clase *JComponent*, que desciende de la clase *Container*. De *JComponent*, los componentes Swing heredan las siguientes funcionalidades:

- **Bordes.** Usando el método *setBorder*, se puede especificar el borde que muestra un componente alrededor de sus lados.
- **Doble buffer.** El doble buffer puede mejorar la apariencia de un componente que cambie frecuentemente. No hay necesidad de escribir el código del doble buffer ya que Swing lo proporciona.

- **Tooltips.** Especificando un string con el método `setToolTipText`, se proporciona ayuda al usuario de un componente. Cuando el cursor se para sobre el componente, el String especificado se muestra en una pequeña ventana que aparece cerca del componente.
- **Navegación con teclado.** Usando el método `registerKeyboardAction`, el usuario puede usar el teclado en vez del ratón para moverse por el GUI.

**JFrame:** Es un contenedor Swing de alto nivel que proporciona ventanas para applets y aplicaciones. Un frame tiene decoraciones como un borde, un título, y botones para cerrar y minimizar la ventana. Un programa típico simplemente crea un frame, añade componentes al panel de contenido, y quizás añade una barra de menú.

**JPanel:** Es un contenedor de propósito general para componentes de peso ligero. Como todos los contenedores, utiliza un Controlador de Distribución para posicionar y dimensionar sus componentes. Como todos los componentes Swing, JPanel permite añadirle bordes y determinar si utiliza el doble buffer para aumentar el rendimiento.

**JTabbedPane:** Con la clase `JTabbedPane`, se puede tener varios componentes (normalmente objetos `JPanel`) compartiendo el mismo espacio. El usuario puede elegir qué componente ver, seleccionando la pestaña del componente deseado.

**JTree:** Con la clase `JTree`, se puede mostrar un árbol de datos. `JTree` realmente no contiene datos, simplemente es un vista de ellos. `JTree` muestra los datos verticalmente. Cada fila contiene exactamente un ítem de datos (llamado un nodo).

Cada árbol tiene un nodo raíz (llamado `Root` en la figura anterior, del que descienden todos los nodos. Los nodos que no pueden tener hijos se llaman nodos leaf (hoja).

**JScrollPane:** Un `JScrollPane` proporciona una vista desplazable de un componente ligero. Cuando el estado de la pantalla real está limitado, se utiliza un `ScrollPane` para mostrar un componente que es grande o cuyo tamaño puede cambiar dinámicamente.

**JSplitPane:** Un `JSplitPane` contiene dos componentes de peso ligero, separados por un divisor. Arrastrando el divisor, el usuario puede especificar qué cantidad de área pertenece a cada componente. Un `SplitPane` se utiliza cuando dos componentes contienen información relacionada y se quiere que el usuario pueda cambiar el tamaño de los componentes en relación a uno o a otro.

**JTable:** Con la clase JTable, se pueden mostrar tablas de datos, y opcionalmente permitir que el usuario los edite. JTable no contiene ni almacena datos; simplemente es una vista de los datos.

**JFileChooser:** La clase JFileChooser proporciona un UI para elegir un fichero de una lista. Un selector de ficheros es un componente que se puede situar en cualquier lugar del GUI. La clase JFileChooser hace sencillo traer un dialogo modal que contiene un selector de ficheros.

**JTextArea:** Es un área de texto que muestra múltiples líneas de texto y permite que el usuario edite el texto con el teclado y el ratón.

**JRadioButton:** Son grupos de botones en los que, por convención, solo uno de ellos puede estar seleccionado, exhibiendo de esta manera su estado al usuario.

**JComboBox:** Es un componente con el que el usuario puede teclear un valor o elegirlo desde una lista. Un ComboBox editable ahorra tiempo de entrada proporcionando atajos para los valores más comúnmente introducidos.

**JSpinner:** Esta clase provee en una sola línea la posibilidad al usuario de entrar o seleccionar un valor de una secuencia pedida. Los JSpinner proporcionan generalmente un par de los botones minúsculos de flechas para cambiar entre los elementos de la secuencia.

## **4. ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE LA HERRAMIENTA STARCUBE**

En el desarrollo de este proyecto se diseñó e implementó una herramienta ROLAP para el apoyo a toma de decisiones a las PYMES. En STARCUBE se implementa la conexión con bodegas de datos, el modelamiento multidimensional para creación de la estructura de los cubos de datos, además de la manipulación y visualización de los cubos mediante las operaciones de OLAP.

Para lograr a cabo este cometido fue necesario el estudio de varias herramientas OLAP para la inteligencia de negocios existentes en el mercado, para determinar debilidades y fortalezas de las mismas y de esta manera poder desarrollar una herramienta en la que se integraran todas las fortalezas vistas y se eliminaran las dificultades encontradas.

El desarrollo de la herramienta de código abierto bajo software libre y con metodología orientada a objetos hace posible la reutilización de código y permite acoplar funcionalidades con las que da la posibilidad de incluir nuevas características a la herramienta contribuyendo al mejoramiento de la misma.

La interfaz gráfica de la misma permite una interacción amigable entre el usuario y la herramienta, facilitando la visualización e interpretación de los análisis.

Para la construcción de la herramienta se desarrollaron: un módulo de utilidades para realizar la conexión con la base de datos y que además contiene todas las clases y bibliotecas que son utilizadas en toda la aplicación, un módulo de Modelamiento Multidimensional para la creación de los cubos de datos y un módulo que permita visualizar y manipular de manera gráfica cubos de datos.

A continuación puede apreciarse la descripción formal del desarrollo del proyecto STARCUBE.

### **4.1 ANÁLISIS DE STARCUBE**

Para el análisis de STARCUBE, se construyen los siguientes artefactos:

- Tabla de funciones o requerimientos.
- Casos de uso.
- Diagramas de casos de uso.



#### 4.1.1 Funciones

En la tabla 2.1, se describen los requerimientos de la herramienta STARCUBE.

Tabla 2.1. Funciones de STARCUBE

Ref#	Función	Cat	Atributo	Detalles y restricciones	Cat
R1	Ejecutar la Herramienta	Evidente	Interfaz		Obligatorio
R1.1	Desplegar ventana de Login	Evidente	Interfaz		Obligatorio
R1.2	Mostrar interfaz gráfica de la herramienta	Evidente	Interfaz		Obligatorio
R2	Mostrar mensajes de ayuda, respuesta o error cuando sea necesario	Evidente	Interfaz	Cuadros de Diálogo	Deseable
R3	Permitir Crear Cubos	Evidente	Interfaz		Opcional
R3.1	Autorizar conexiones a Bases de Datos	Evidente	Interfaz		Obligatorio
R3.2	Permitir Seleccionar Tablas	Evidente	Interfaz	Seleccionar 1 tabla de hechos por cubo y las tablas dimensiones deben tener integridad referencial con est.	Opcional
R3.3	Permitir la selección de medidas.	Evidente	Interfaz	Seleccionar campos que	Opcional
R3.4	Permitir la creación de Dimensiones	Evidente	Interfaz		Opcional
R3.5	Permitir Guardar	Evidente	Interfaz		Opcional

	Cubos	ente			
R4	Permitir Eliminar Cubos	Evid ente	Interfaz		Opcional
R5	Manipular Cubos	Evid ente	Interfaz		Opcional
R5.1	Permitir Operación Dice	Evid ente	Interfaz		Opcional
R5.2	Permitir Operación Slice	Evid ente	Interfaz		Opcional
R5.3	Permitir Operación Pivot	Evid ente	Interfaz		Opcional
R5.4	Permitir Operación Drill Down	Evid ente	Interfaz		Opcional
R5.5	Permitir Operación RollUp	Evid ente	Interfaz		Opcional
R6	Manejar Usuarios	Evid ente	Interfaz		Opcional
R6.1	Crear Usuarios	Evid ente	Interfaz		Opcional
R6.2	Editar Usuarios	Evid ente	Interfaz		Opcional
R6.3	Eliminar Usuarios	Evid ente	Interfaz		Opcional

#### 4.1.2 Casos de uso

Los casos de Uso de la herramienta software STARCUBE se presentan desde la tabla 2.2 hasta la tabla 2.6.

Tabla 2.2. Caso de uso inicio

<b>Caso de Uso: Inicio</b>
1. El Usuario Ejecuta la aplicación.
2. El Sistema le presenta una ventana de logueo.
3. El Usuario se loguea.
4. El sistema carga los elementos necesarios para ejecutar la aplicación y muestra la en otra ventana las opciones que puede seguir el usuario, dependiendo del tipo de usuario el sistema le permitirá escoger que va a hacer.
5. Si es Administrador, el puede escoger crear usuarios, crear cubos,

manejar cubos.
6. Si no es administrador (usuario normal), el sistema le permitirá sólo la manipulación de cubos.
<b>Casos Alternos</b>
<ul style="list-style-type: none"> <li>• Si no se ha inicializado la Base de Datos, el sistema muestra un mensaje de alerta y no permite que se ejecute la aplicación.</li> <li>• Si el usuario que ejecuta la aplicación no está registrado, el sistema presenta un mensaje de aviso y no permite que se ejecute la aplicación.</li> </ul>

Tabla 2.3. Caso de uso manejo de usuarios

<b>Caso de uso manejo de usuarios</b>
1. El administrador ingresa a la opción Usuarios en la aplicación.
2. La aplicación muestra las opciones de: Agregar Nuevo Usuario, Editar Usuario, Eliminar Usuario.
3. El administrador escoge la opción de Agregar nuevo usuario.
4. El sistema indica al administrador las opciones para que registre el nuevo usuario.
5. El administrador registra al nuevo usuario y le asigna, o no, privilegios de administrador.
6. El sistema exhibe al administrador un mensaje de éxito en la operación y vuelve a la pantalla inicial donde se muestran las tres opciones para el manejo de usuarios.
7. El administrador escoge la opción Editar Usuario.
8. El sistema muestra la lista de usuarios existentes.
9. El administrador escoge el usuario a editar.
10. El sistema indica los datos pertenecientes al usuario a editar.
11. El administrador edita los datos que él necesita y presiona el botón de aceptación.
12. El sistema almacena la nueva información del usuario a editar, exhibe un mensaje de éxito y vuelve a la pantalla inicial donde se indican las tres opciones para el manejo de usuarios.
13. El administrador escoge la opción Eliminar Usuario.
14. El sistema le presenta la lista de usuarios registrados en el sistema.
15. El administrador escoge el usuario a eliminar y presiona el botón de eliminar usuario.
16. El sistema borra los datos del usuario a eliminar y actualiza la lista de usuarios.
<b>Casos Alternos</b>
<ul style="list-style-type: none"> <li>• Si un usuario que no tiene privilegios de administrador quiere ingresar a la opción de Usuarios en la aplicación, el sistema no permite ejecutar la opción.</li> </ul>

<ul style="list-style-type: none"> <li>• Si el administrador registra un nuevo usuario el cual ya existe, el sistema le presentará un mensaje, avisándole al administrador.</li> </ul>
<ul style="list-style-type: none"> <li>• Si al editar usuarios el administrador escribió el nombre de un usuario que ya estaba registrado, el sistema muestra un mensaje al administrador avisándole del hecho y no permite guardar cambios.</li> </ul>

Tabla 2.4. Caso de uso creación de cubos

<b>Caso de uso creación de cubos</b>
1. El administrador presiona el botón de crear cubo.
2. El sistema muestra en un asistente de creación de cubos el primer paso: selección de fuente de datos.
3. El administrador configura la fuente de datos y presiona el botón siguiente.
4. El sistema exhibe en el asistente el segundo paso: escoger las tablas de la base de datos.
5. El administrador escoge las tablas que necesitará para crear su cubo y presiona el botón siguiente.
6. El sistema muestra en el asistente el tercer paso: seleccionar las medidas.
7. El administrador selecciona las medidas que necesite y presiona el botón siguiente.
8. El sistema le presenta en el asistente el cuarto paso: seleccionar las dimensiones.
9. El administrador selecciona las dimensiones y crea tantas jerarquías como necesite y presiona el botón siguiente.
10. El sistema le presenta el final del asistente: almacenar cubo.
11. El administrador asigna los privilegios del cubo, lo guarda y finaliza el asistente.

Tabla 2.5. Caso de uso manipulación de cubos

<b>Caso de uso manipulación de cubos</b>
▪ El usuario escoge la opción de Abrir Cubos.
▪ El sistema le presenta los cubos existentes en la base de datos.
▪ El usuario escoge el cubo que necesita.
▪ El sistema abre la ventana de navegación por el cubo.
▪ El usuario navega por el cubo y empieza a armar su análisis con las dimensiones que necesite y presiona el botón de comienzo de análisis.
▪ El sistema realiza sus procesos correspondientes y le al usuario la tabla y la gráfica correspondiente del análisis escogido, además de las todas las

operaciones para que interactúe con el análisis.
<b>Casos Alternos</b>
<ul style="list-style-type: none"> <li>El usuario presiona la opción de abrir cubos y el administrador no los ha creado, el sistema muestra una ventana vacía, sin cubos.</li> </ul>

Tabla 2.6. Caso de uso operaciones

<b>Caso de uso operaciones</b>
1. El Usuario presiona el botón de comenzar análisis.
2. El sistema muestra al usuario la tabla y la gráfica correspondientes al análisis, además activa las opciones de operaciones que se pueden realizar sobre el análisis Dice, Drill Down, Roll Up, Pivot, Manejo de Measures.
3. El usuario escoge la opción Dice.
4. El sistema le presenta al usuario la lista de elementos de otras dimensiones para que realice la restricción necesaria.
5. El usuario escoge el (los) elemento(s) para realizar la restricción al análisis.
6. El sistema inmediatamente cambia los datos en la tabla y en la gráfica.
7. El usuario escoge la opción de realizar Pivot.
8. El sistema realiza un intercambio de ejes en la tabla de datos, lo que inmediatamente se ve reflejado en la gráfica.
9. El usuario decide navegar por las estructuras jerárquicas de las dimensiones, presionando alguno de los elementos de la cabecera de la tabla o de la izquierda de la misma (donde están las dimensiones correspondientes a las filas).
10. El sistema realiza la operación y modifica la tabla y la gráfica.
11. El usuario presiona el botón de medidas.
12. El sistema le presenta la liste de las medidas del cubo.
13. El usuario escoge qué medida(s) necesita para mejorar su análisis.
14. El sistema realiza los procesos necesarios internos y modifica la tabla y la gráfica.

#### 4.1.3 Diagramas de casos de uso

Los diagramas de casos uso de la herramienta STARCUBE, se muestran en la figuras 2.11 hasta la figura 2.15

Figura 2.11. STARCUBE

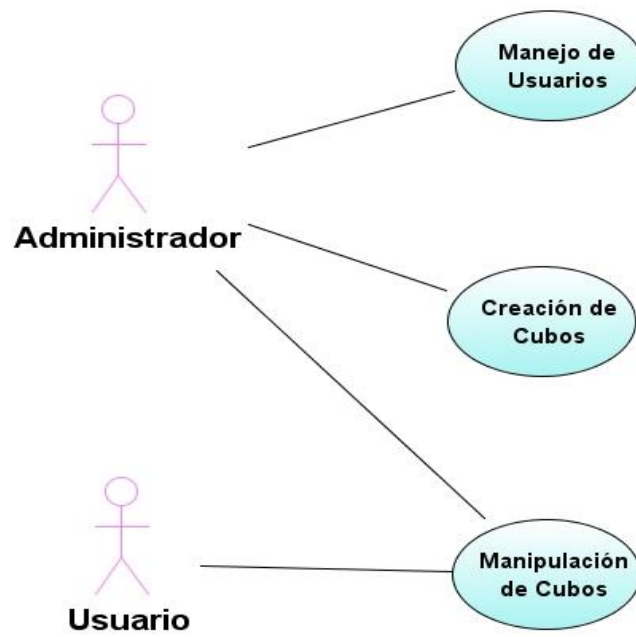


Figura 2.12. Manejo de usuarios

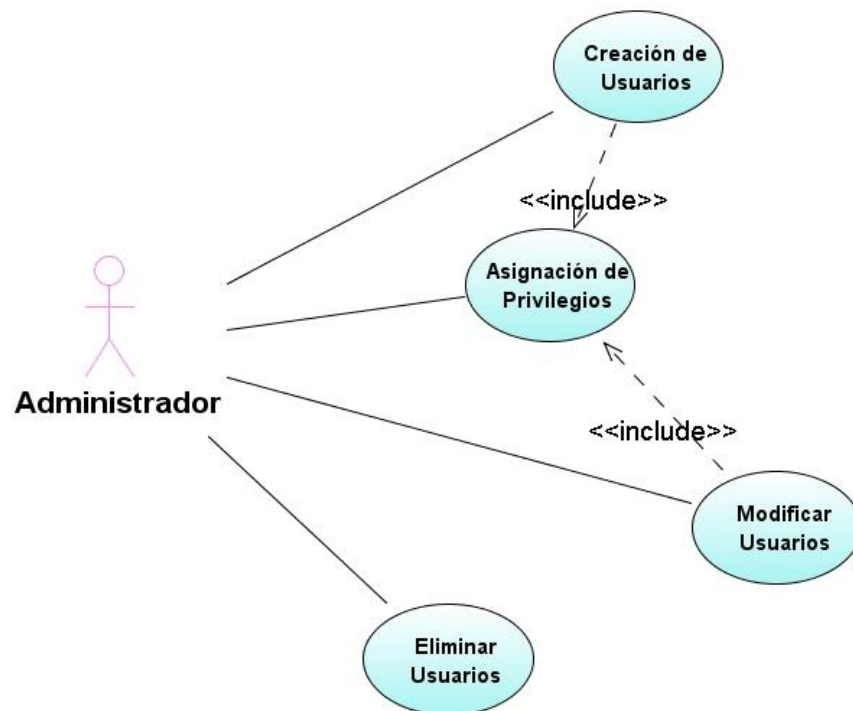


Figura 2.13. Creación de cubos

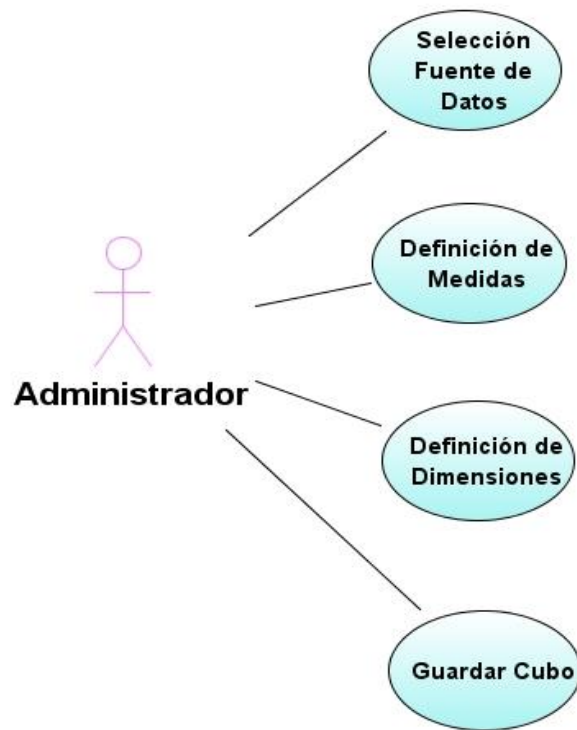


Figura 2.14. Manipulación de cubos

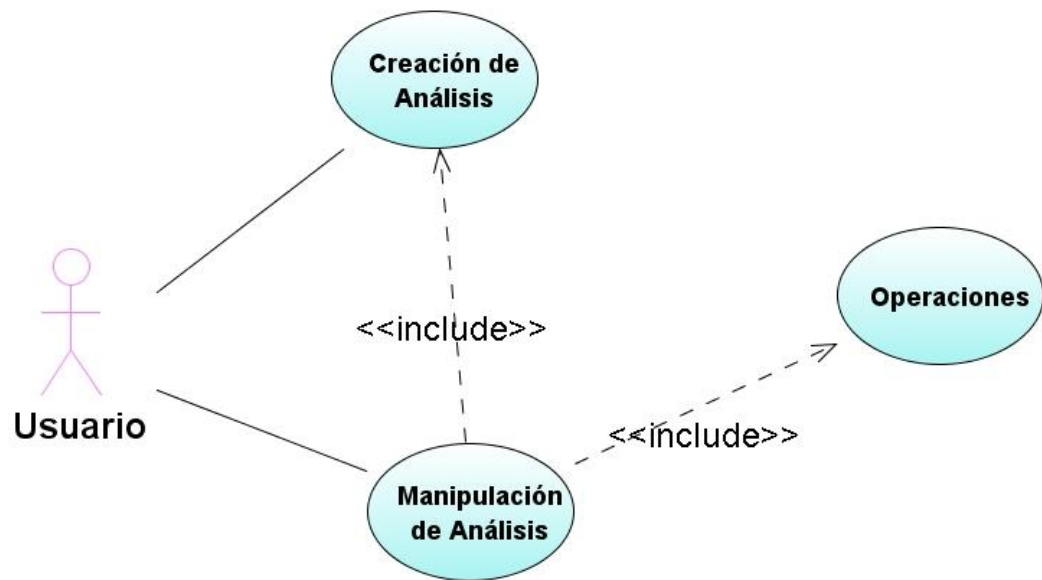
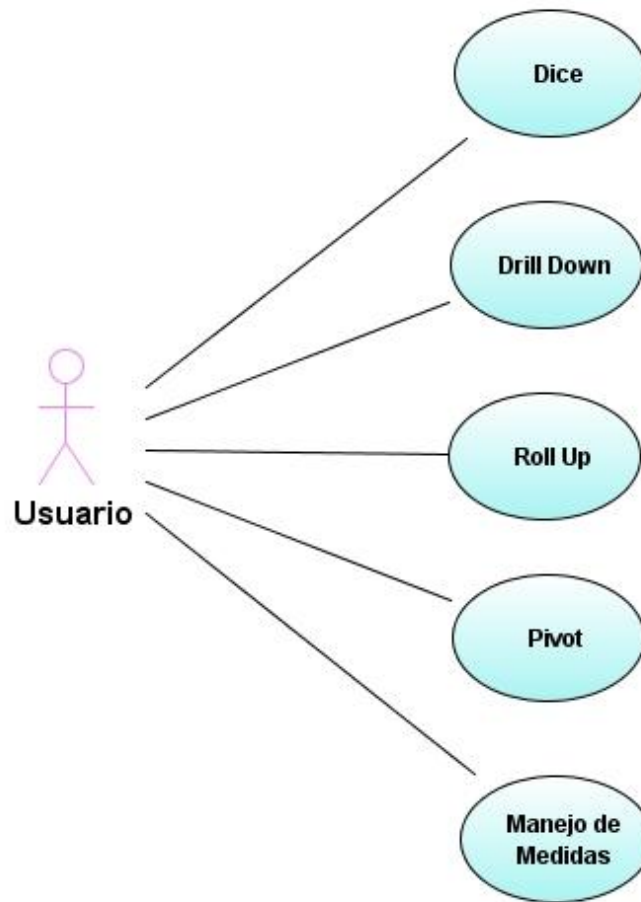


Figura 2.15. Operaciones



## 4.2 DISEÑO DE STARCUBE

Para el diseño de STARCUBE se hace uso de los siguientes artefactos:

- Diagramas de Secuencia.
- Diagramas de Paquetes.
- Diagrama de Clases

### 4.2.1 Diagramas de secuencia

Los diagramas de secuencia de la herramienta se presentan desde la figura 2.16 hasta la figura 2.21.



Figura 2.16. Add measure table



Figura 2.17. Delete measure table

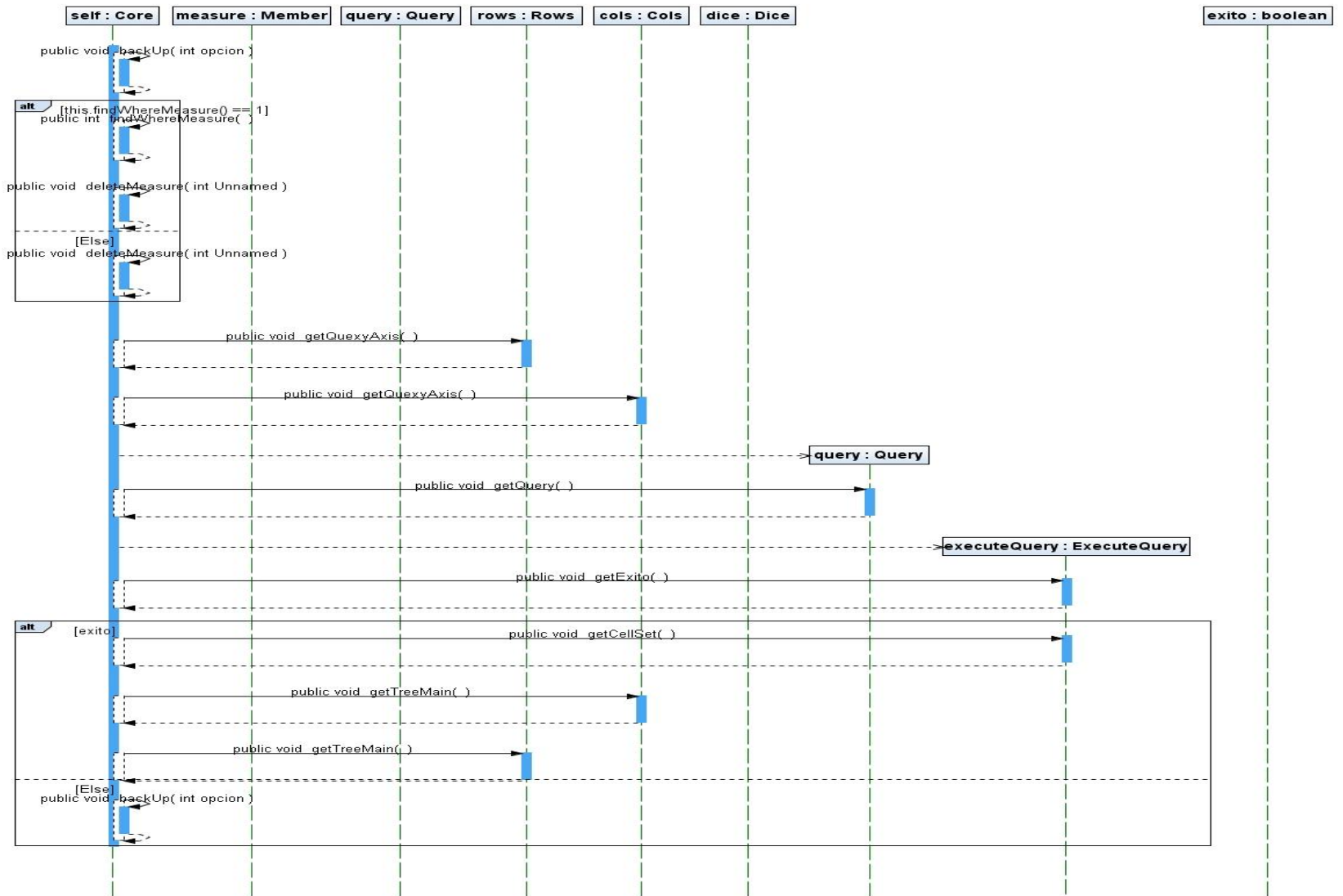


Figura 2.18. Delete member

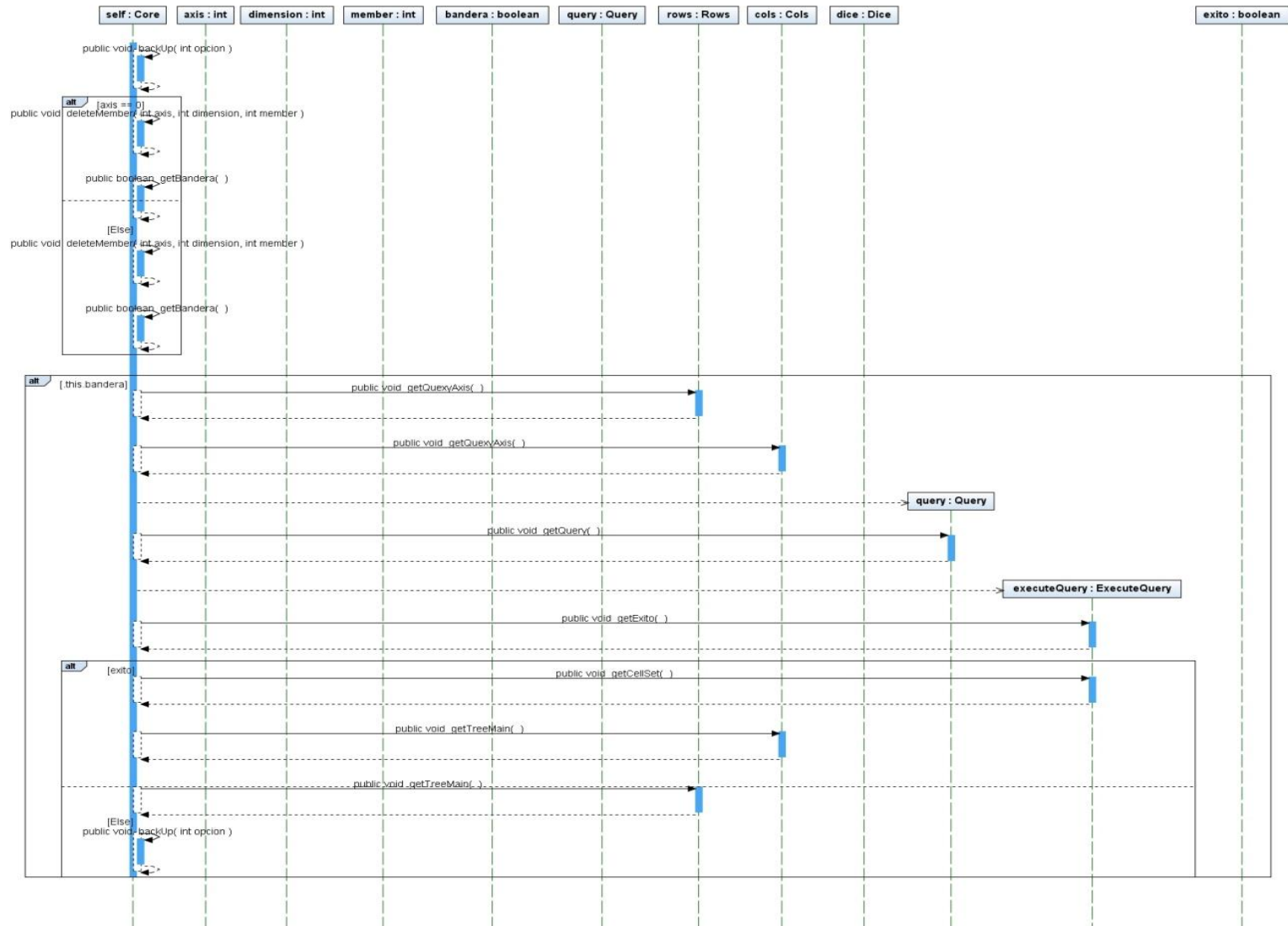


Figura 2.19. Operación pivot

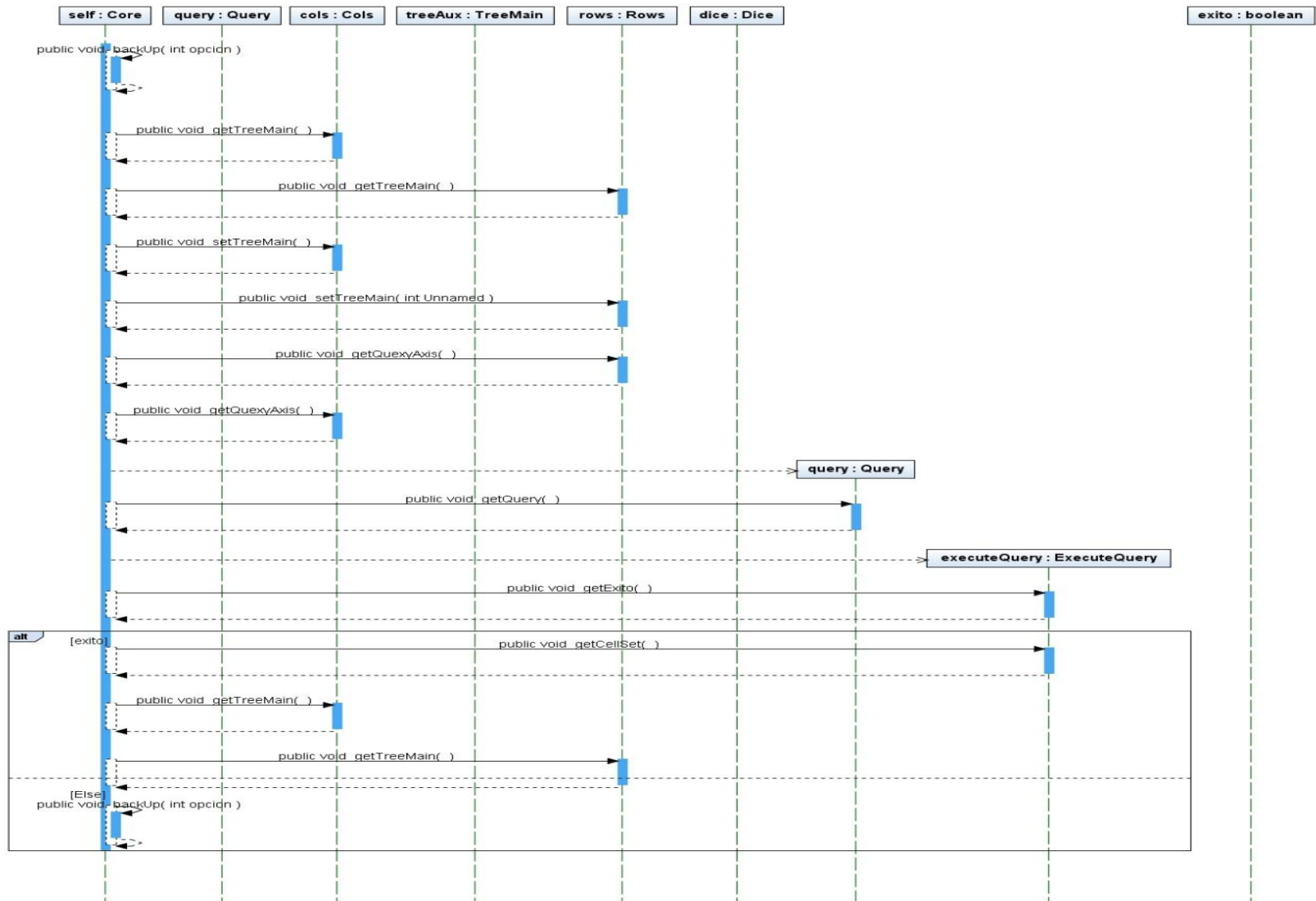
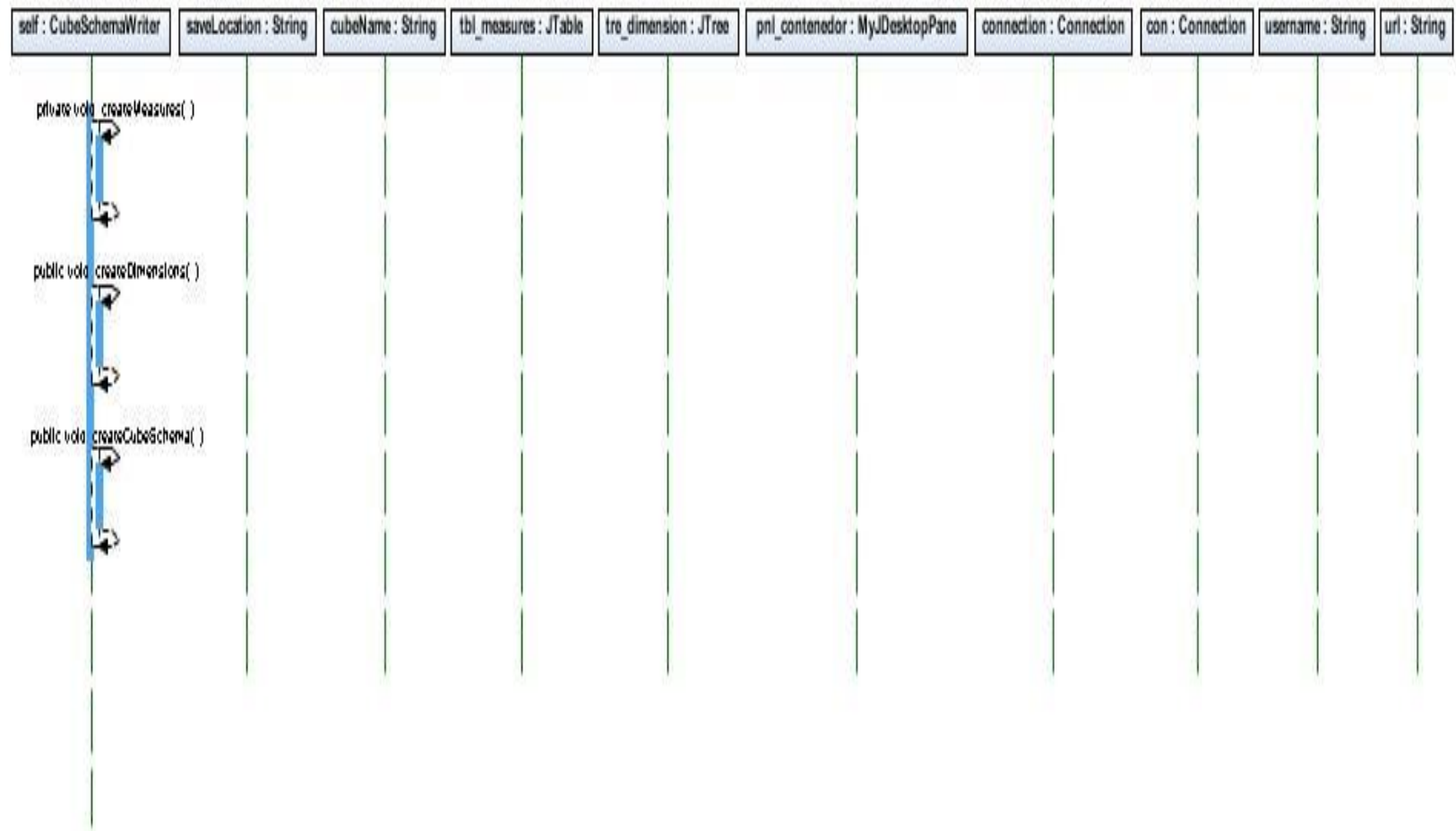


Figura 2.20. Drill-down, roll-up



Figura 2.21. Configuración y creación del esquema del cubo



#### 4.2.2 Diagramas de paquetes

El diseño de los diagramas de paquetes de la herramienta STARCUBE se muestra en las figuras 2.22, figura 2.23 y figura 2.24.

Figura 2.22. Paquete principal

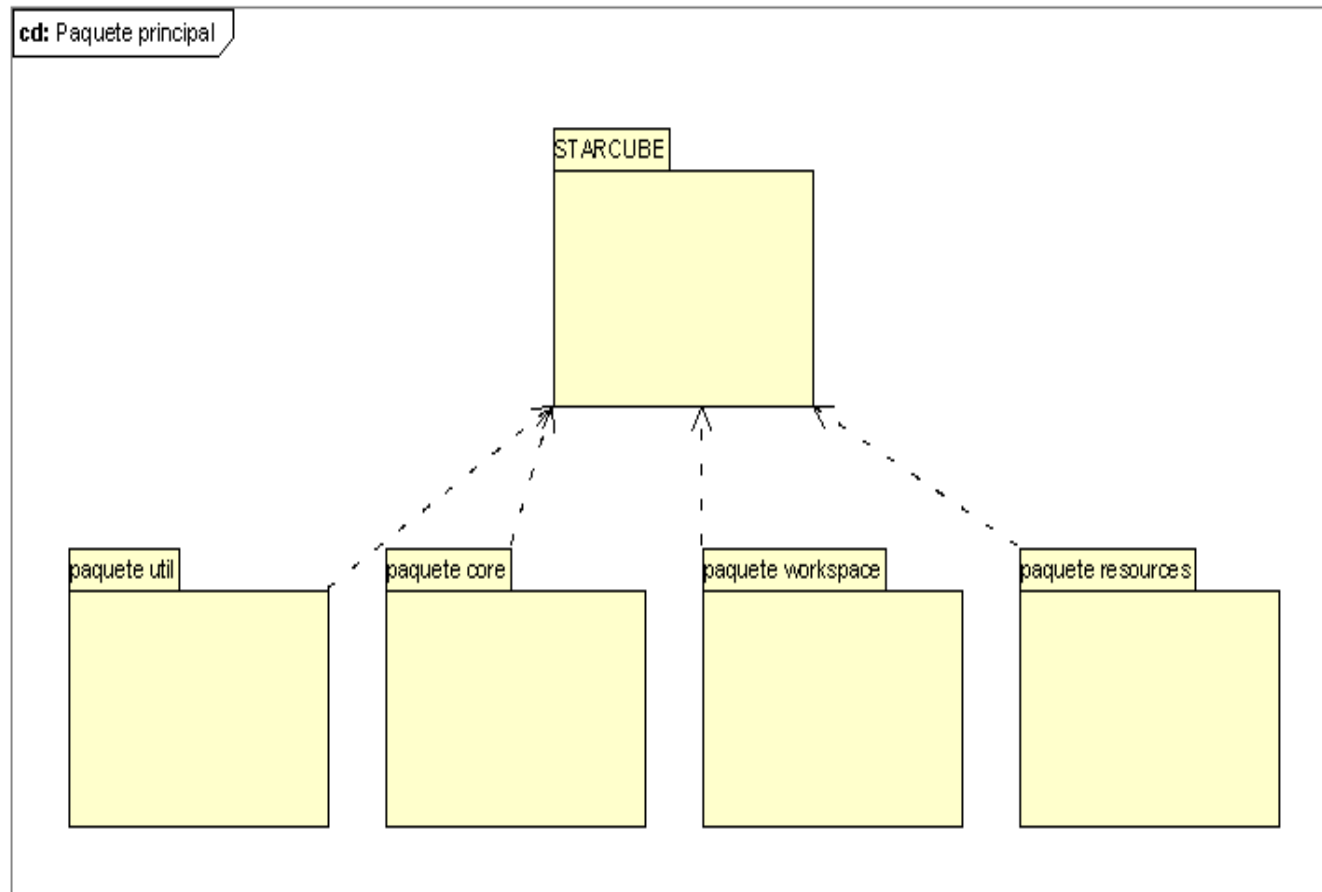


Figura 2.23. Package core

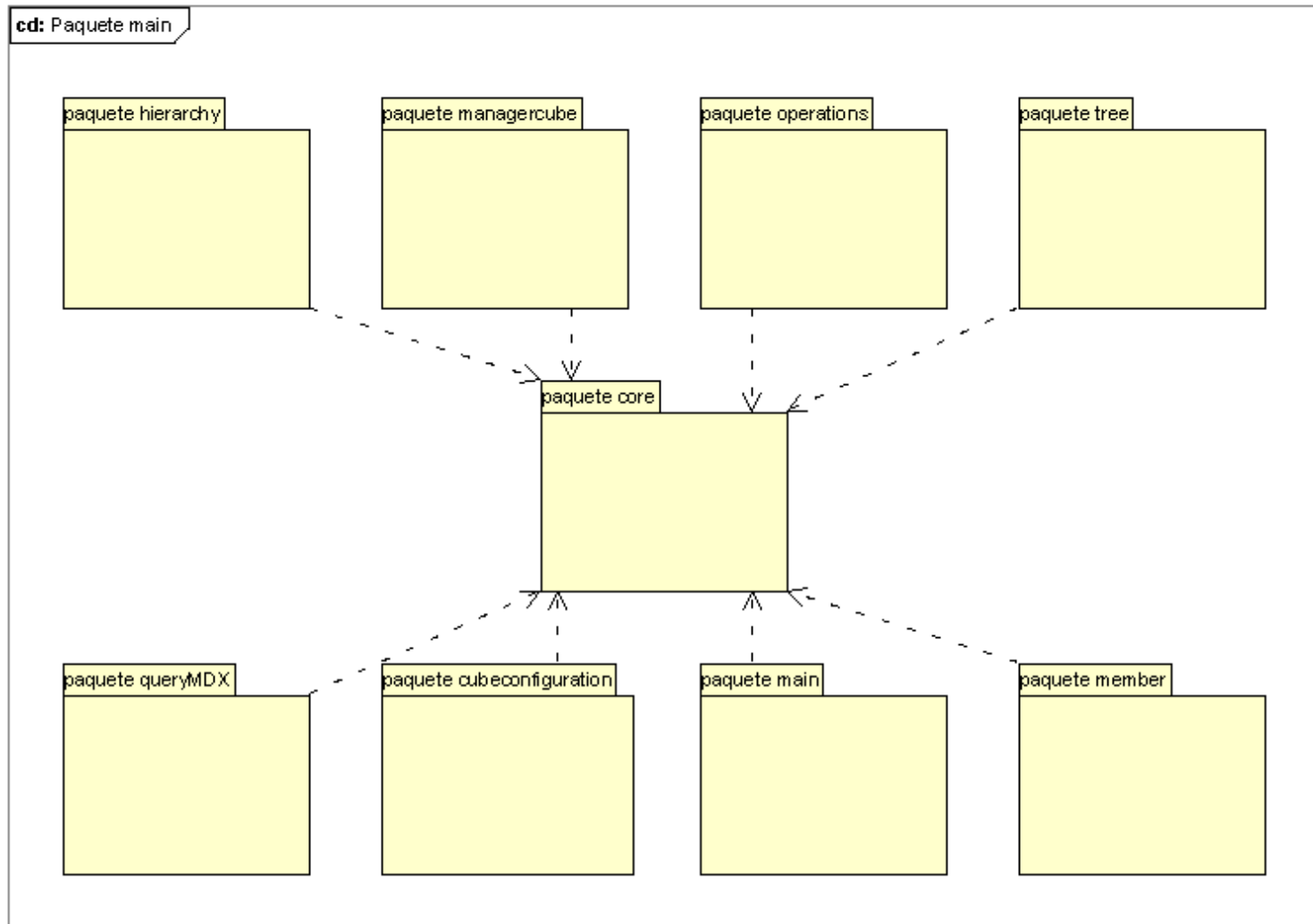
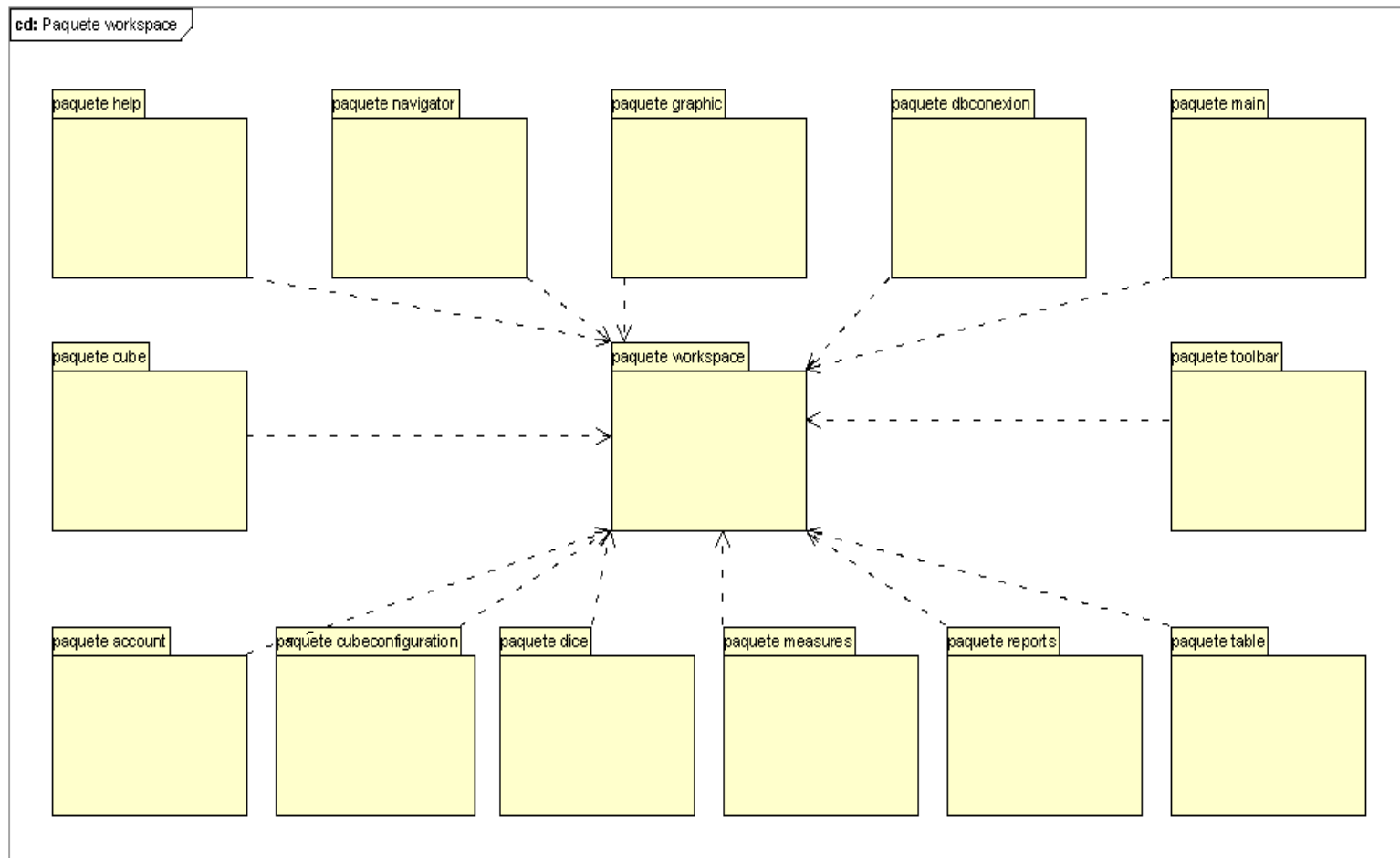




Figura 2.24. Paquete interfaz gráfica

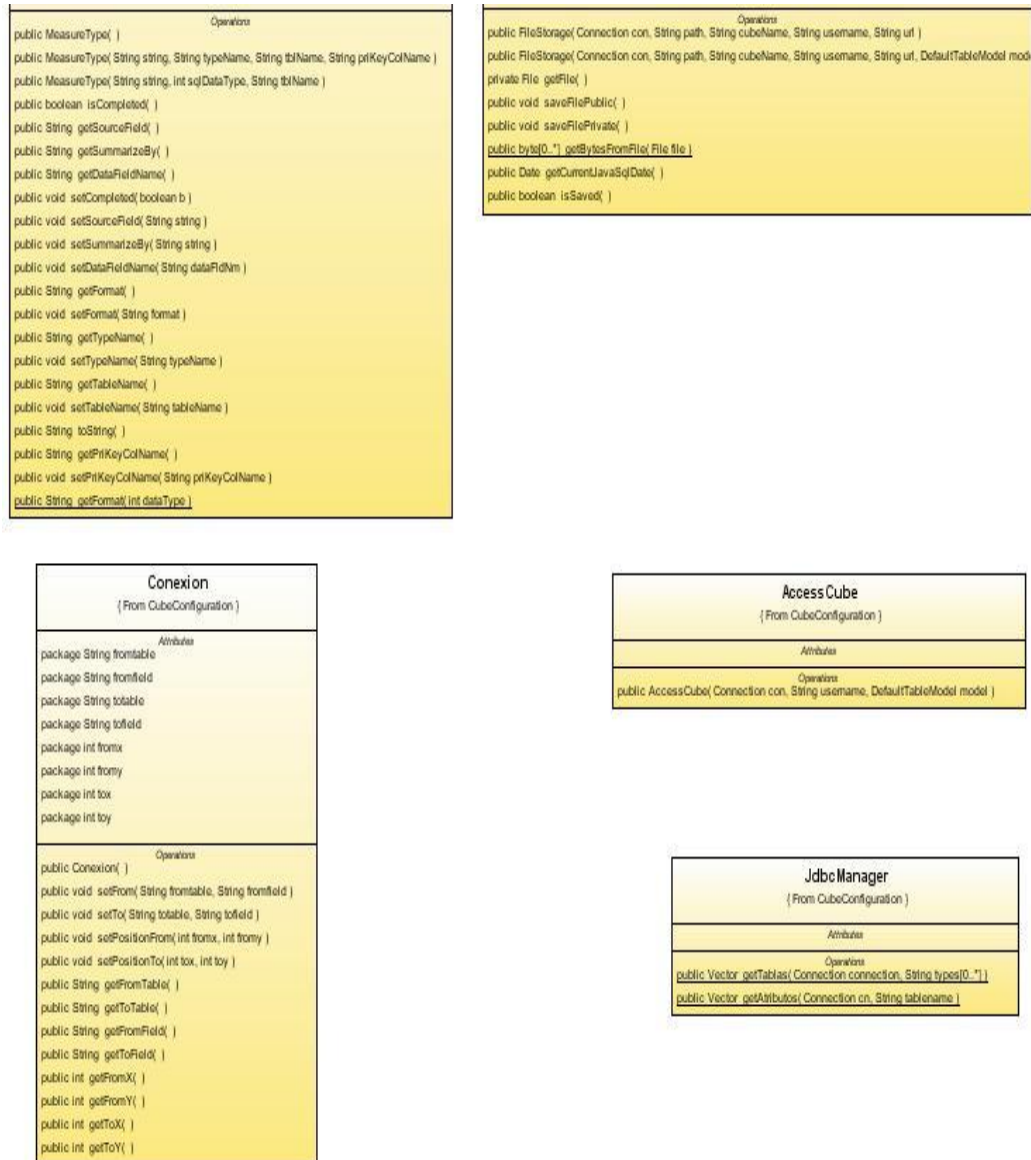


### 4.2.3 Diagramas de clases STARCUBE

El diseño de los diagramas de clases de la herramienta STARCUBE se muestran en la figuras 2.25 hasta la figura 2.44.

#### Paquete Core

Figura 2.25. Cubeconfiguration



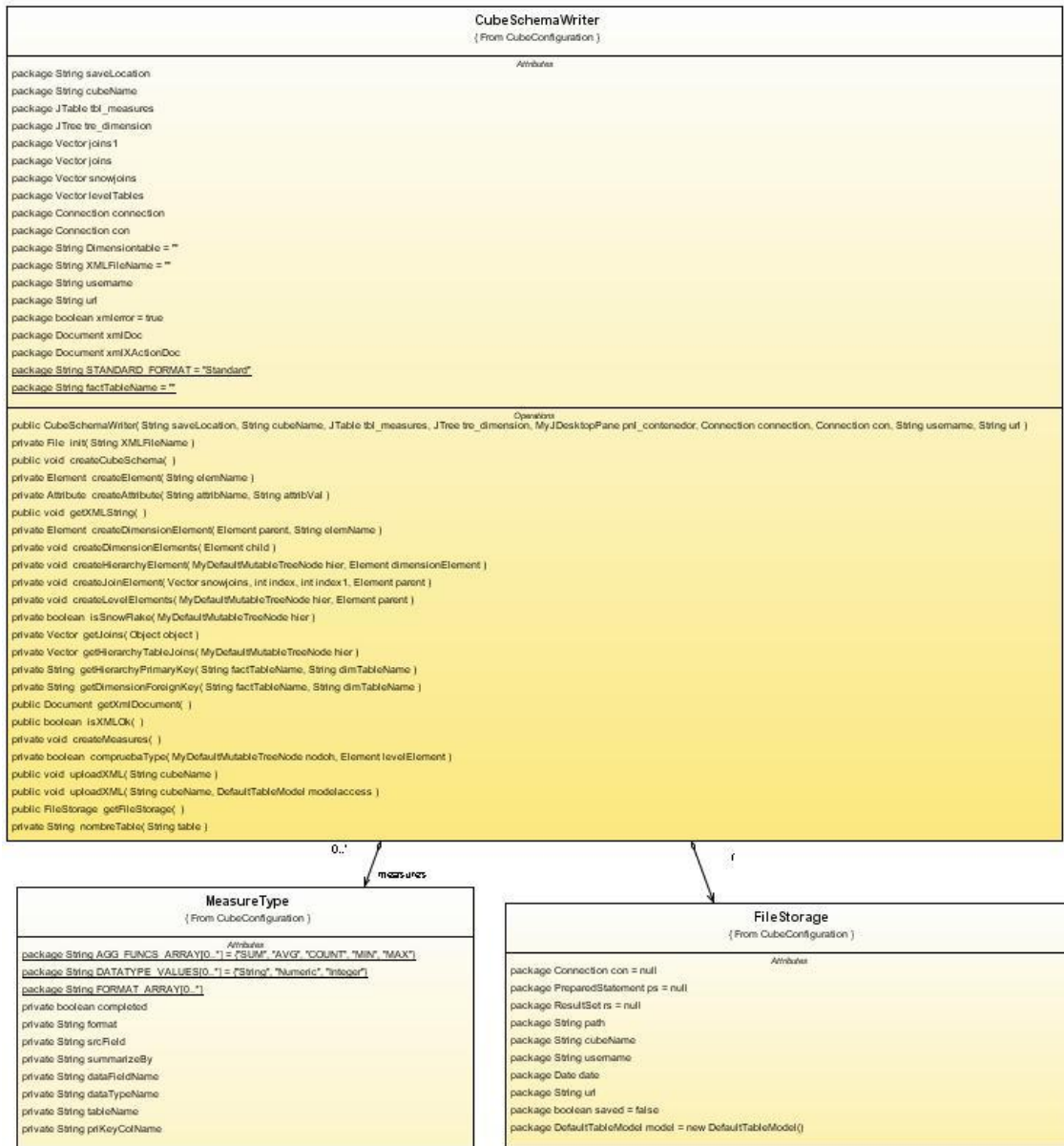


Figura 2.26. Hierarchy



Figura 2.27. Main

<<interface>> <b>Core</b> { From main }
<i>Atributos</i>
<pre> public OlapConnection connection public CellSet cellset package boolean bandera = true package boolean exito = true package String cubeName package Member membersMeasures[0..*] package Vector backUpCols package Vector backUpRows </pre>
<i>Operations</i>
<pre> public Core( Vector row, Vector columns, OlapConnection connection, String cubeName public Vector createMembers( Vector members ) public Vector createHierarchies( Vector hierarchies ) public void drillDown_RollUp( int axis, int dimension, int member ) public void pivot( ) public void dice( Vector elementosDice ) public void deleteMember( int axis, int dimension, int member ) public void backUp( int opcion ) public int findWhereMeasure( ) public Vector findWhatmeasure( int opcion ) public void alterMeasureDice( Member newMeasure ) public void deleteMeasureTable( Member measure ) public void addMeasureTable( Member measure ) public void addAllMeasureTable( ) public Vector getVectorMainCol( ) public Vector getVectorMainRow( ) public CellSet getCellSet( ) public TreeMain getTreeRow( ) public TreeMain getTreeCol( ) public boolean getBandera( ) public Vector getVectorHierarchiesCols( ) public Vector getVectorHierarchiesRows( ) public boolean getExito( ) </pre>

Figura 2.28. Managercube

<b>Manager</b> { From ManagerCube }
<i>Attributes</i>
package JScrollPane pane1 package JScrollPane paneG package JPanel mdx package JPanel paneDice package JPanel panelMeasures package String cubename = "" package boolean band = true
<i>Operations</i>
public Manager( JScrollPane pane1, JScrollPane paneG, JPanel mdx, JPanel paneDice, JPanel panelMeasures public void setCore( Core core ) public Core getCore( ) public JScrollPane getPanel( ) public JScrollPane getPanelG( ) public JPanel getPanelDice( ) public JPanel getPanelMeasures( ) public void setCubeName( String cubename ) public void setFixedT( SpanTable fixedT ) public void setDataT( SpanTable dataT ) public String getCubeName( ) public void printAll( ) public TableOlap getMainTable( )

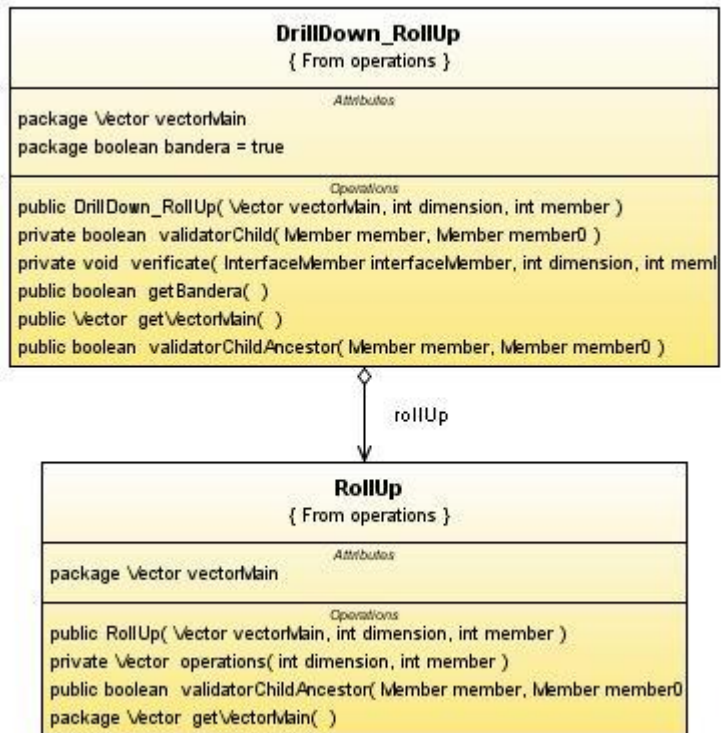
Figura 2.29. Member

<b>ManagementMembers</b> { From member }
<i>Attributes</i>
package \vector vectorTree package \vector vectorMain
<i>Operations</i>
public ManagementMembers( \vector vectorMembers ) private void create\vectorMembers( ) public \vector get\vectorMain( ) public void set\vectorMain( \vector vectorMain )

<b>InterfaceMember</b> { From member }
<i>Attributes</i>
package \vector vectorMain
<i>Operations</i>
public InterfaceMember( \vector vectorMain ) public \vector get\vectorDimension( int dimension ) public MyNodeMember getMyNodeMember( int dimension, int index ) public int getNumDimension( ) public int getNumMember( int dimension )

<b>MyNodeMember</b> { From member }
<i>Attributes</i>
package int father = 0 package Member member = null
<i>Operations</i>
public MyNodeMember( Member member ) public Member getMember( ) public int getFather( ) public void setFather( int father ) public void setMember( Member member ) public Object clone( )

Figura 2.30. Operations



<b>Slice</b> { From operations }
Attributes
Operations

<b>DeleteMember</b> { From operations }
Attributes package boolean bandera = true package Vector vectorMain
Operations public DeleteMember( Vector vectorMain, int dimension, int member ) public boolean getBandera( ) public Vector getVectorMain( ) private Vector operations( int dimension, int member )

<b>BorrarPrueba</b> { From operations }
Attributes package Vector vector package int opcion private Choice choice1
Operations public BorrarPrueba( Vector vector, int opcion ) private void haber( ) private void initComponents( )

<b>Dice</b> { From operations }
Attributes public Vector elementosDice package String queryDice
Operations public Dice( Vector elementosDice ) public String getQueryDice( ) private void crearQueryDice( ) public Vector getElementosDice( ) public void setElementoDice( Member elementoNuevo )

<b>DrillDown</b> { From operations }
Attributes package Vector vectorMain package boolean bandera = true
Operations public DrillDown( Vector vectorMain, int dimension, int member ) public boolean getBandera( ) public Vector getVectorMain( ) private Vector operations( int dimension, int member ) private Vector operations2( int dimension, int member ) public void imprimirVector( Vector vector )

<b>DeleteMeasureTable</b> { From operations }
Attributes package Vector vectorMain
Operations public DeleteMeasureTable( Vector vectorMain, Member measure ) private Vector operations( Member measure ) public Vector getVectorMain( )

<b>Pivot</b> { From operations }
Attributes
Operations

<b>AddMeasureTable</b> { From operations }
Attributes package Vector vectorMain
Operations public AddMeasureTable( Vector vectorMain, Member measure ) private Vector operations( Member measure ) public Vector getVectorMain( ) private Vector operations2( Member measure )



Figura 2.31. Querymdx

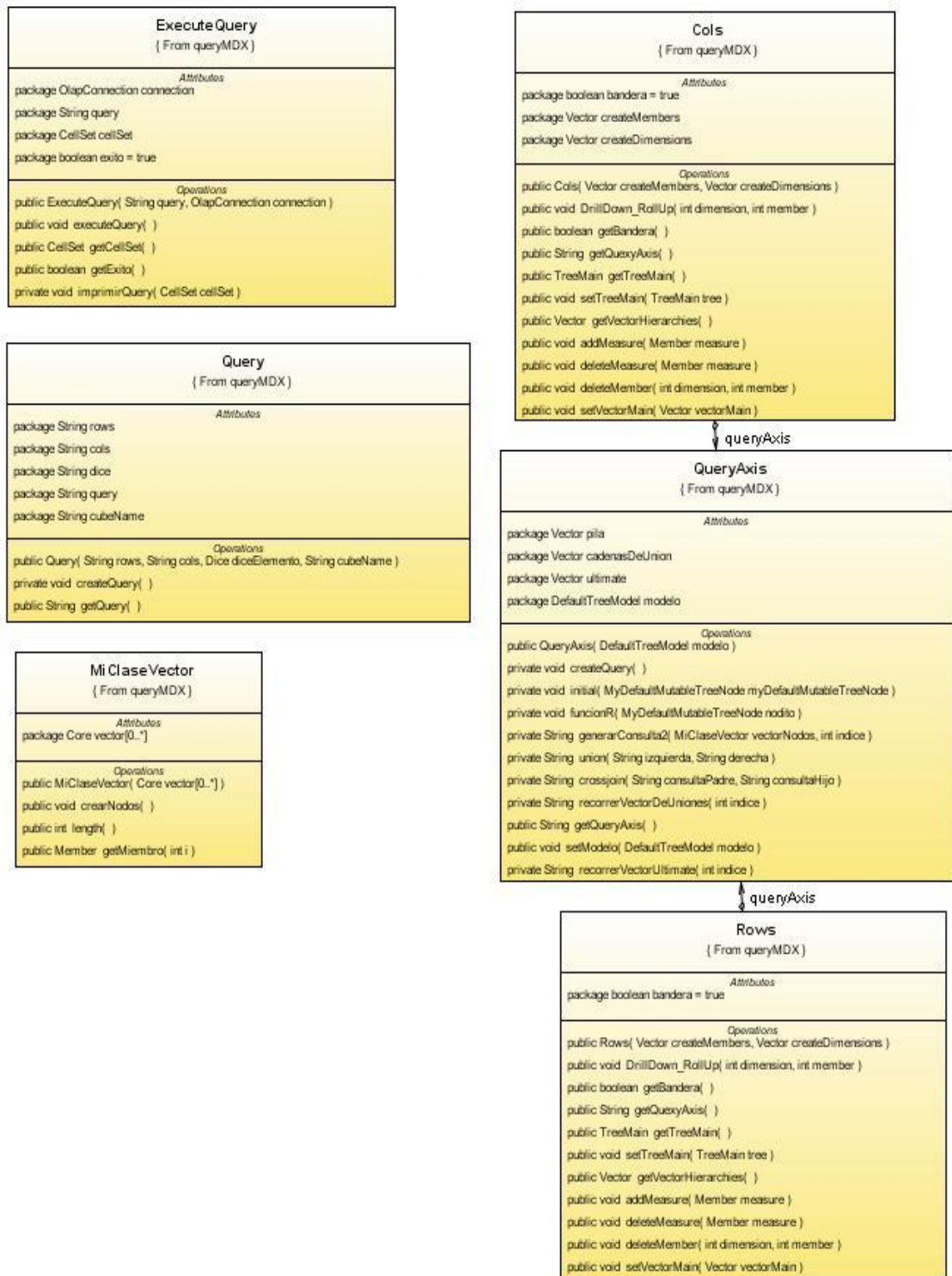


Figura 2.32. Tree

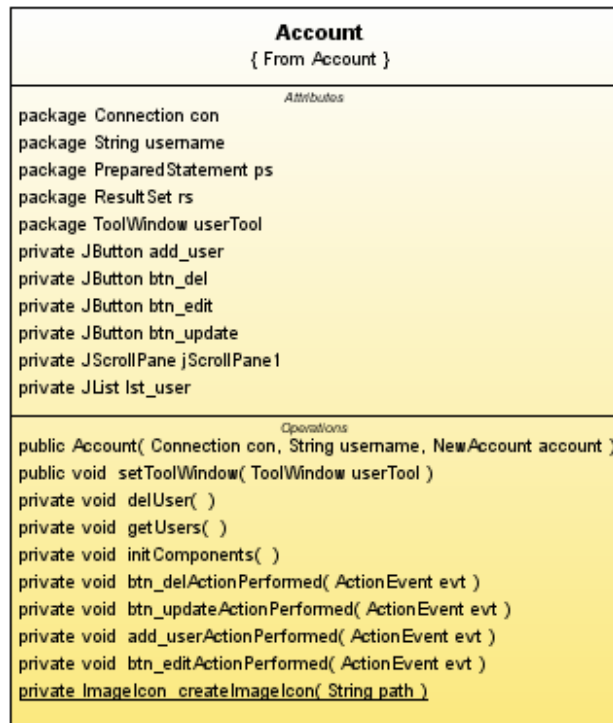
<b>TreeMain</b> { From tree }
<i>Attributes</i>
<pre>package DefaultTreeModel modelo package boolean bandera = true package Vector createMembers package Vector createDimensions</pre>
<i>Operations</i>
<pre>public TreeMain( Vector createMembers, Vector createDimensions ) public void deleteMember( int dimension, int member ) public Vector getVectorMain( ) private DefaultTreeModel createTree( Vector vectorMain ) private Vector createMyDefaultTreeNodes( Vector vectorMain ) public DefaultTreeModel getModelo( ) public void imprimirArbol( MyDefaultMutableTreeNode nodo ) public void imprimirVectorMyDefaultMutableTreeNode( Vector vecto public void drillDown_RollUp( int dimension, int member ) public void addMeasure( Member measure ) public void deleteMeasure( Member measure ) public boolean getBandera( ) public Vector getVectorHierarchy( ) public void setVectorMain( Vector vectorMain )</pre>


<b>MyDefaultMutableTreeNode</b> { From tree }
<i>Attributes</i>
<pre>package Member miembro package int lugarJerarquia = 0 package int father = 0</pre>
<i>Operations</i>
<pre>public MyDefaultMutableTreeNode( Member miembro public String getName( ) public String getUniqueName( ) public Member getMiembro( ) public int getLugarJerarquia( ) public int getFather( ) public void setLugarJerarquia( int lugarJerarquia ) public void setFather( int father )</pre>

## Paquete Workspace

Figura 2.33. Account

<b>MyCellRendererUser</b> { From Account }
<i>Attributes</i>
<pre>package ImageIcon icono = createImageIcon("resources/bossuser.png")</pre>
<i>Operations</i>
<pre>public MyCellRendererUser( ) private ImageIcon createImageIcon( String path )</pre>
<i>Operations Redefined From ListCellRenderev</i>
<pre>public Component getListCellRendererComponent( JList list, Object value, int index, boolean isSelected, boolean cellHasFocus</pre>



 account

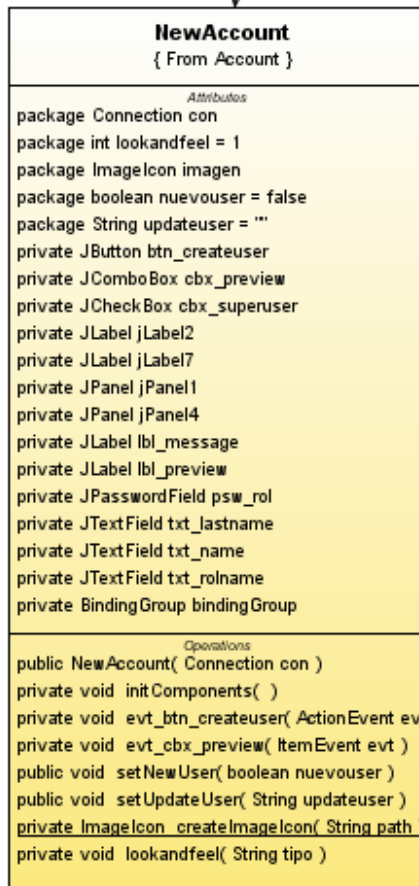
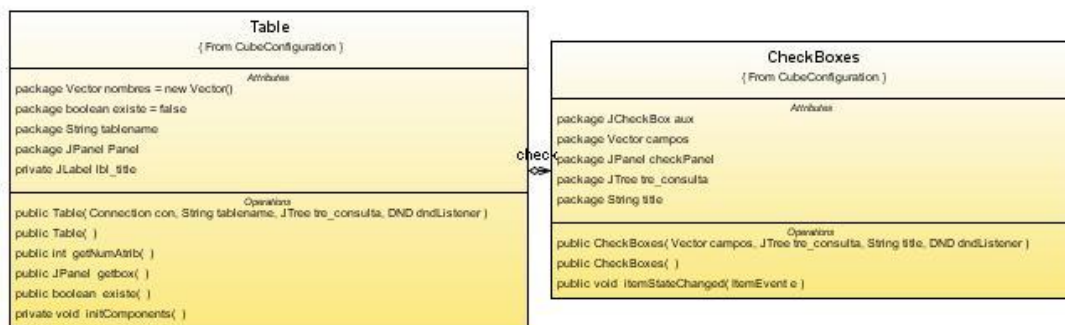


Figura 2.34. Cube



Figura 2.35. Cubeconfiguration



<b>MyTreeSelectionListener</b> { From Cube Configuration }
<i>Attributes</i>
package DefaultTableModel model
<i>Operations</i>
public MyTreeSelectionListener( DefaultTableModel model ) public void valueChanged( TreeSelectionEvent e )

<b>MyTreeModelListener</b> { From Cube Configuration }
<i>Attributes</i>
<i>Operations</i>
public void treeNodesChanged( TreeModelEvent e ) public void treeNodesInserted( TreeModelEvent e ) public void treeNodesRemoved( TreeModelEvent e ) public void treeStructureChanged( TreeModelEvent e )

<b>MyDefaultMutableTreeNode</b> { From Cube Configuration }
<i>Attributes</i>
package String name = null package String namecolumn = null package String uniquenessmembers = null
<i>Operations</i>
public MyDefaultMutableTreeNode( String name, String namecolumn, String uniquenessmembers ) public MyDefaultMutableTreeNode( String name ) public MyDefaultMutableTreeNode( ) public String getName( ) public String getNameColumn( ) public String getUniquenessMembers( ) public void setName( String name ) public void setNameColumn( String namecolumn ) public void setUniquenessMembers( String uniquenessmembers )

<b>DimensionCellRenderer</b> { From Cube Configuration }
<i>Attributes</i>
<i>Operations</i>
public DimensionCellRenderer( ) public Component getTreeCellRendererComponent( JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row, boolean hasFocus ) private ImageIcon createImageIcon( String path )

<b>QueryCellRenderer</b> { From Cube Configuration }
<i>Attributes</i>
<i>Operations</i>
public QueryCellRenderer( ) public Component getTreeCellRendererComponent( JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row, boolean hasFocus ) private ImageIcon createImageIcon( String path )

<b>CubeCellRenderer</b> { From Cube Configuration }
<i>Attributes</i>
<i>Operations</i>
public CubeCellRenderer( ) public Component getTreeCellRendererComponent( JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row, boolean hasFocus )

<b>MyTableListener</b> { From Cube Configuration }
<i>Attributes</i>
package int row package int column package JTree tre_dimension package JTable tbl_dimensionProperties
<i>Operations</i>
public MyTableListener( JTree tre_dimension ) public MyTableListener( JTree tre_dimension, JTable tbl_dimensionProperties ) public void tableChanged( TableModelEvent e )

<b>ShapeStroke</b> { From Cube Configuration }
<i>Attributes</i>
private Shape shapes[0..*] private float advance private boolean stretchToFit = false private boolean repeat = true private AffineTransform t = new AffineTransform() private float FLATNESS = 1
<i>Operations</i>
public ShapeStroke( Shape shapes, float advance ) public ShapeStroke( Shape shapes[0..*], float advance ) public Shape createStrokedShape( Shape shape )

<b>MyMouseAdapter</b> { From Cube Configuration }
<i>Attributes</i>
package DefaultTableModel auxtbl
<i>Operations</i>
public MyMouseAdapter( DefaultTableModel auxtbl ) public MyMouseAdapter( ) public void mouseClicked( MouseEvent e )

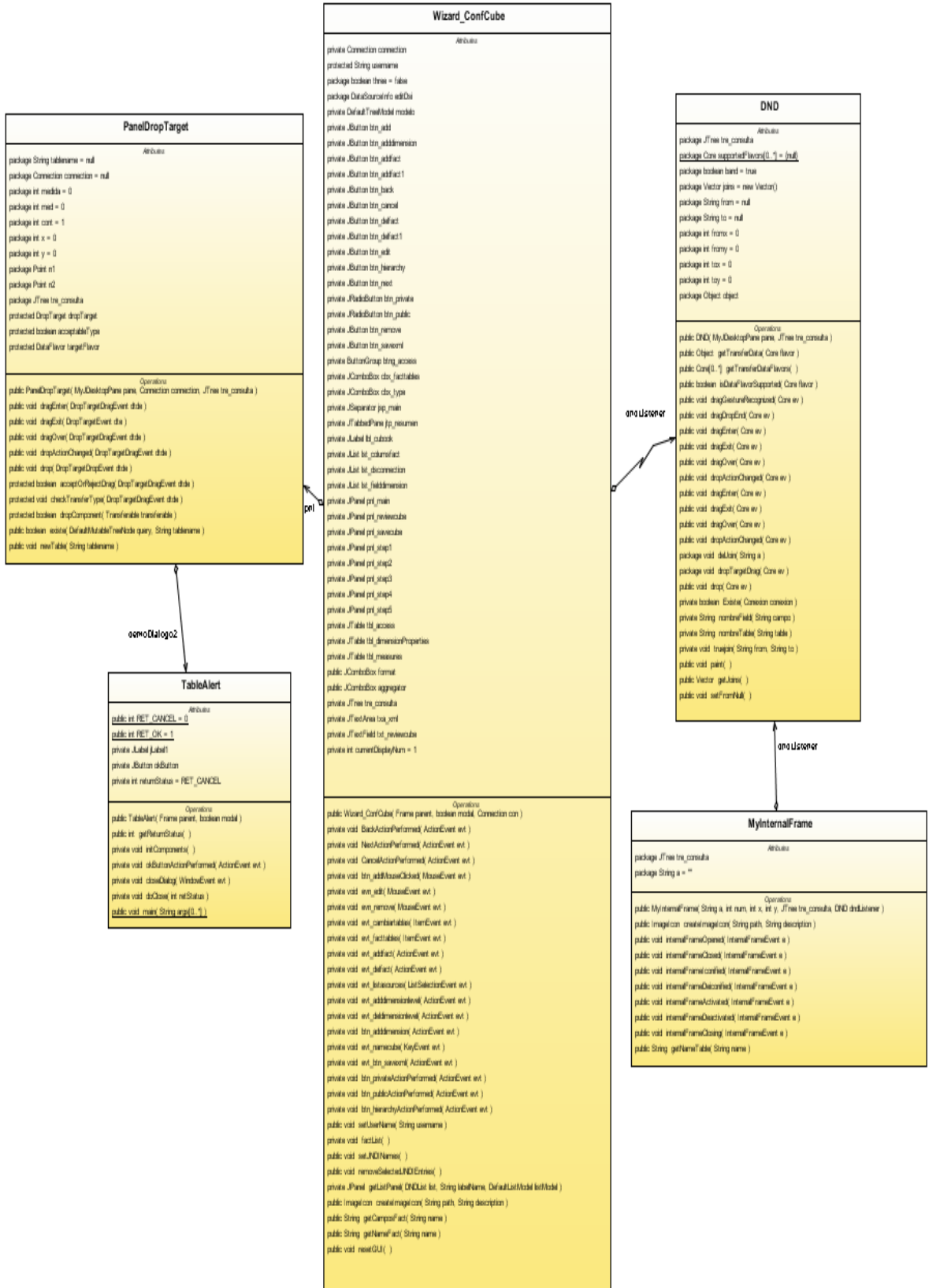


Figura 2.36. Dbconexion

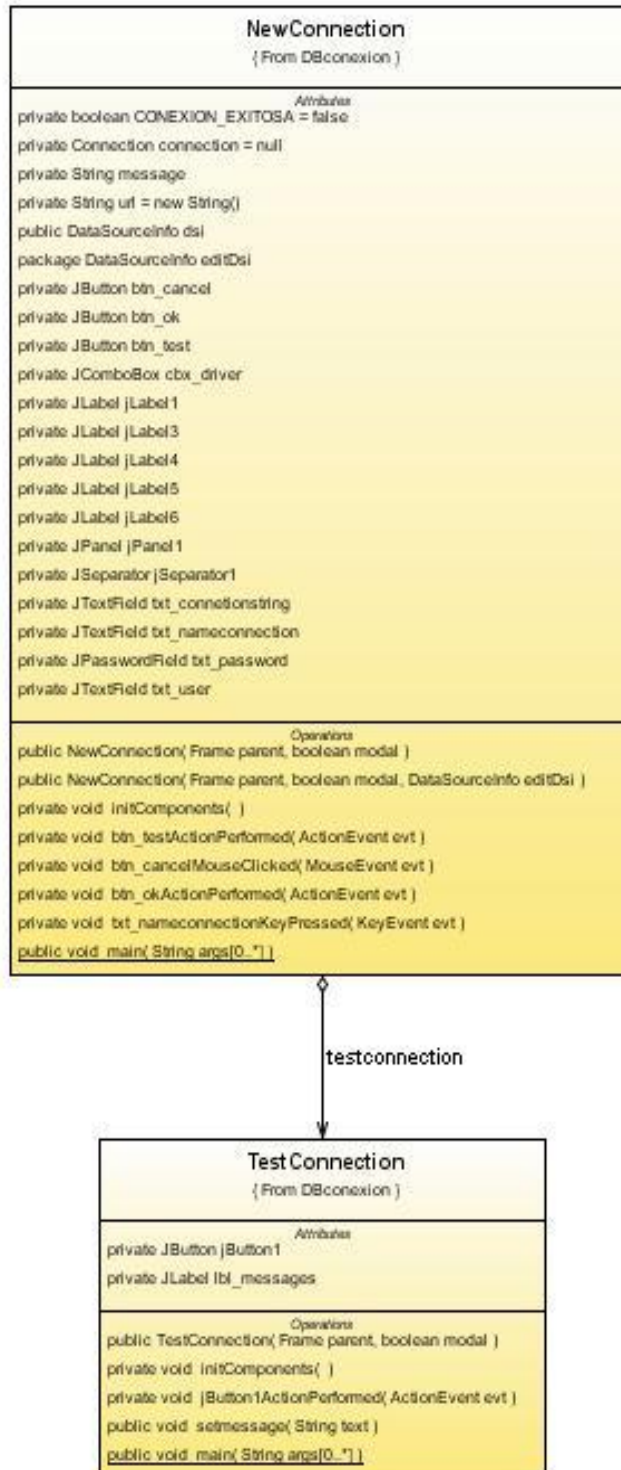


Figura 2.37. Dice

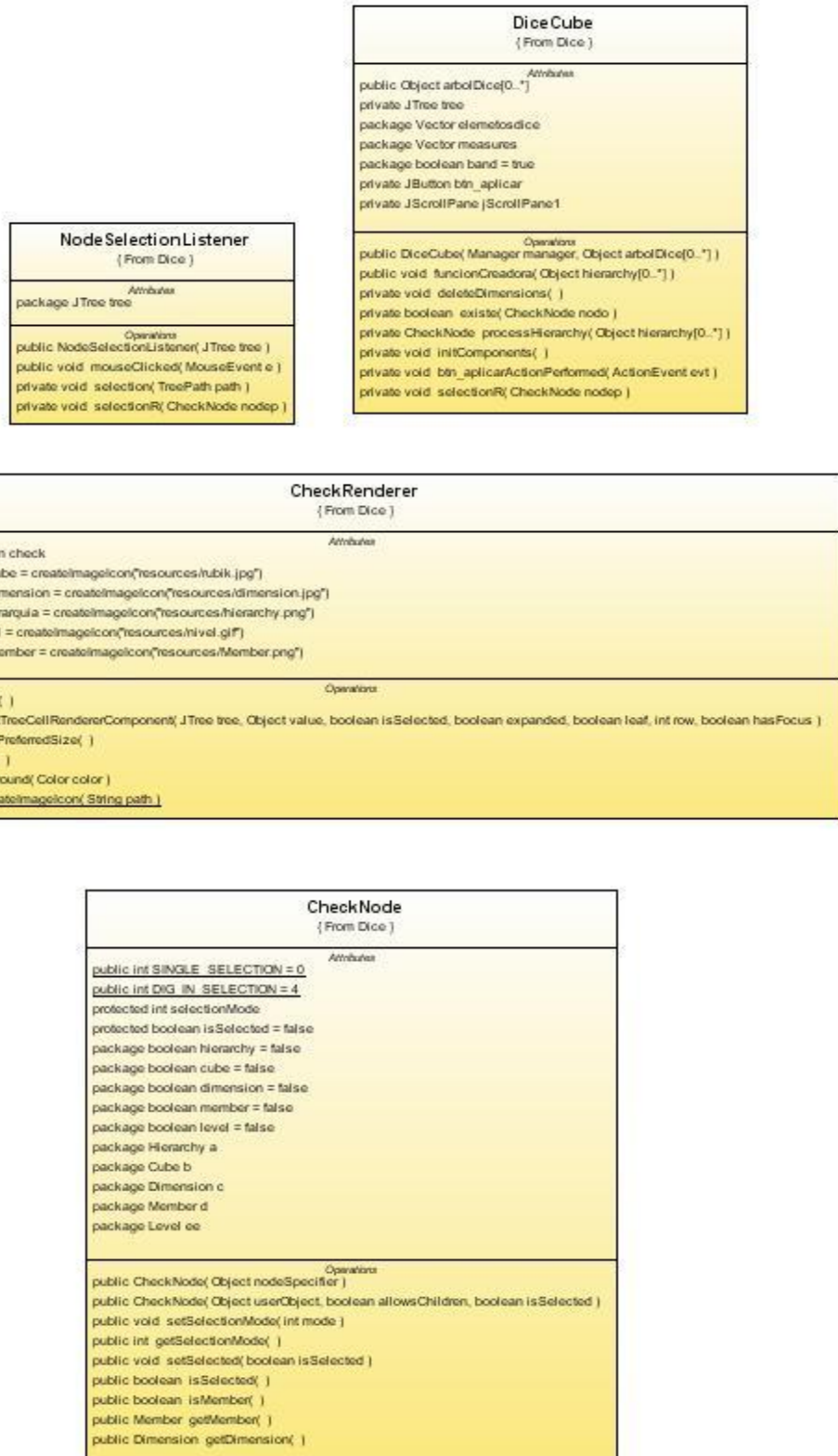




Figura 2.38. Graphic

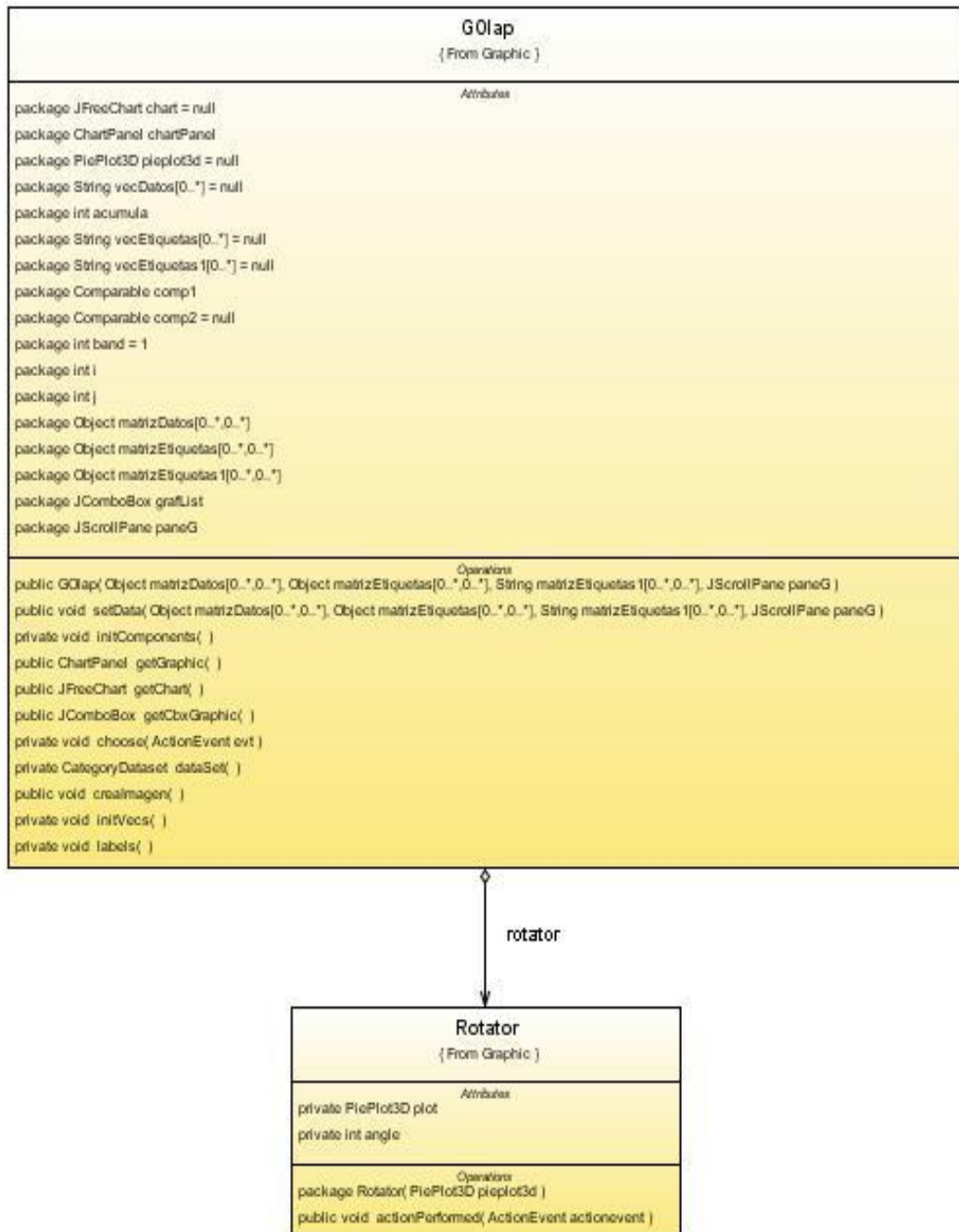


Figura 2.39. Main

<b>MyLookAndFeel</b> ( From main )
<i>Attributes</i>
package JFrame frame <u>public NimRODLookAndFeel nt</u> <u>public NimRODTheme nt</u> package int tip
<i>Operations</i>
public MyLookAndFeel( int tip, JFrame frame ) public MyLookAndFeel( String text, JFrame frame ) public void selectFileTheme( String tip ) public void tip( int tip ) protected void changeTheme( ) public NimRODTheme getTheme( ) public int getTip( ) <u>private InputStream getFile( String path )</u>

<b>StarCubeDoggySet</b> ( From main )
<i>Attributes</i>
public JFrame frame protected ToolWindowManager toolWindowManager protected ViewContext myDoggySetContext protected DockableDescriptor memoryMonitorDescriptor package JXLoginPane paneuser = null package JDBCLoginService svc protected String username package Connection con = null package PreparedStatement ps = null package ResultSet rs = null package int lookandfeel = 8 package int superuser = 1 package JScrollPane pane1 package JScrollPane paneG package JPanel pane package JPanel paneDice package JPanel mdx package JPanel paneMeasures
<i>Operations</i>
public StarCubeDoggySet( ) public void setUp( String username ) public void start( Runnable runnable ) public ToolWindowManager getToolWindowManager( ) public ViewContext getMyDoggySetContext( ) protected void initComponents( ) protected void initMenuBar( ) protected void initToolWindowManager( ) protected void customizeToolWindowManager( MyDoggyToolWindowManager myDoggyToolWindowManager ) protected void dispose( ) <u>public void main( String args[0..*] )</u> public void logeo( ) private void initContentManager( ) protected void setupContentManagerUI( ) <u>private Image createImageIcon( String path )</u> <u>private ImageIcon createIcon( String path )</u>

<b>MyItemListenerLookAndFeel</b> { From main }
<i>Attributes</i>
package JFrame frame package Connection con package String username package JScrollPane paneT
<i>Operations</i>
public MyItemListenerLookAndFeel( JFrame frame, Connection con, String username, JScrollPane paneT ) public void actionPerformed( ActionEvent e ) public MyItemListenerLookAndFeel( ) public void itemStateChanged( ItemEvent e ) private void updateLookAndFeel( int sp )

<b>NimRODUtils</b> { From main }
<i>Attributes</i>
public Color rolColor package int THIN = 0 package int FAT = 1 package int MATRIX_FAT = 5 package Kernel kernelFat package int MATRIX_THIN = 3 package Kernel kernelThin
<i>Operations</i>
package NimRODUtils( ) package Color getSombra( ) package Color getBrillo( ) package Color getSombraMenu( ) package Color getBrilloMenu( ) package NimRODTheme : iniCustomColors( NimRODTheme nimrodtheme, String s, String s1, String s2, String s3, String s4, String s5, String s6, String s7, String s8, String s9, String s10, String s11 ) package byte[] readStream( InputStream inputStream ) package void printBarraMenu( Graphics g, JMenuItem[] menuItems, Color color ) package void paintFocus( Graphics g, int i, int j, int k, int l, int i1, int j1, Color color ) package void paintFocus( Graphics g, int i, int j, int k, int l, int i1, int j1, float f, Color color ) package Color getRollOverColor( ) package Color getColorAifa( Color color, int i ) package Color getColorMedio( Color color, Color color1 ) package ColorUIResource getColorTercio( Color color, Color color1 ) private int propInt( int i, int j, int k ) package void paintShadowTitleFat( Graphics g, String s, int i, int j, Color color ) package void paintShadowTitleFat( Graphics g, String s, int i, int j, Color color, Color color1 ) package void paintShadowTitleFat( Graphics g, String s, int i, int j, Color color, Color color1, int k ) package void paintShadowTitleThin( Graphics g, String s, int i, int j, Color color ) package void paintShadowTitleThin( Graphics g, String s, int i, int j, Color color, Color color1 ) package void paintShadowTitleThin( Graphics g, String s, int i, int j, Color color, Color color1, int k ) package void paintShadowTitleFatV( Graphics g, String s, int i, int j, Color color ) package void paintShadowTitleFatV( Graphics g, String s, int i, int j, Color color, Color color1, int k ) package void paintShadowTitleThinV( Graphics g, String s, int i, int j, Color color ) package void paintShadowTitleThinV( Graphics g, String s, int i, int j, Color color, Color color1 ) package void paintShadowTitleThinV( Graphics g, String s, int i, int j, Color color, Color color1, int k ) package void paintShadowTitle( Graphics g, String s, int i, int j, Color color, Color color1, int k, int i1 ) package Icon reescala( Icon icon, int i, int j ) package int getOpacity( ) package int getMenuOpacity( ) package float getMenuOpacityFloat( ) package int getFrameOpacity( ) package float getFrameOpacityFloat( )

Figura 2.40. Measures

<b>MyRadioButton</b> { From Measures }
<i>Attributes</i> package Member member
<i>Operations</i> public MyRadioButton( Object nodeSpecifier ) public Member getMeasure( )

<b>Measures</b> { From Measures }
<i>Attributes</i> package \vector measures package JPanel checkPanel
<i>Operations</i> public Measures( ) public Measures( Manager manager ) private int cuantos( ) private void initComponents( ) public void actionPerformed( ActionEvent e

Figura 2.41. Navigator

```

ListTransferHandler
{ From Navigator }

Attributes
private int indices[0..*] = null
private int addIndex = -1
private int addCount = 0

Operations
public boolean canImport( TransferSupport info )
protected Transferable createTransferable( JComponent c )
public int getSourceActions( JComponent c )
public boolean importData( TransferSupport info )
protected void exportDone( JComponent c, Transferable data, int action )
protected String exportString( JComponent c )
protected void importString( JComponent c, String str )
protected void cleanup( JComponent c, boolean remove )
    
```

```

MyListSelectionListener
{ From Navigator }

Attributes
package JList column

Operations
public MyListSelectionListener( JList column )
public void valueChanged( ListSelectionEvent e )
    
```

```

MyCellRenderer
{ From Navigator }

Attributes

Operations
public MyCellRenderer( )
private ImageIcon createImageIcon( String path )

Operations Inherited From ListCellRenderer
public Component getListCellRendererComponent( JList list, Object value, int index, boolean isSelected, boolean cellHasFocus )
    
```

```

NodesTransferable
{ From Navigator }

Attributes
package DefaultMutableTreeNode nodes[0..*]
private DataFlavor flavors[0..*]
public String NODES_MIME_TYPE = DataFlavor( javaVMLocalObjectMimeType + ".class=javax.swing.tree.DefaultMutableTreeNode" )

Operations
public NodesTransferable( DefaultMutableTreeNode nodes[0..*] )
public Object getTransferData( DataFlavor flavor )
public DataFlavor[0..*] getTransferDataFlavors( )
public boolean isDataFlavorSupported( DataFlavor flavor )
    
```

```

CubeCellRenderer
{ From Navigator }

Attributes

Operations
public CubeCellRenderer( )
public Component getTreeCellRendererComponent( JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row, boolean hasFocus )
private ImageIcon createImageIcon( String path )
    
```

```

MyDefaultMutableTreenodo
{ From Navigator }

Attributes
package boolean hierarchy = false
package boolean cube = false
package boolean dimension = false
package boolean member = false
package boolean level = false
package Hierarchy a
package Cube b
package Dimension c
package Member d
package Level ee

Operations
public MyDefaultMutableTreenodo( Object nodeSpecifier )
public boolean isHierarchy( )
public boolean isCube( )
public boolean isDimension( )
public Hierarchy getHierarchy( )
public Cube getCube( )
public Dimension getDimension( )
    
```

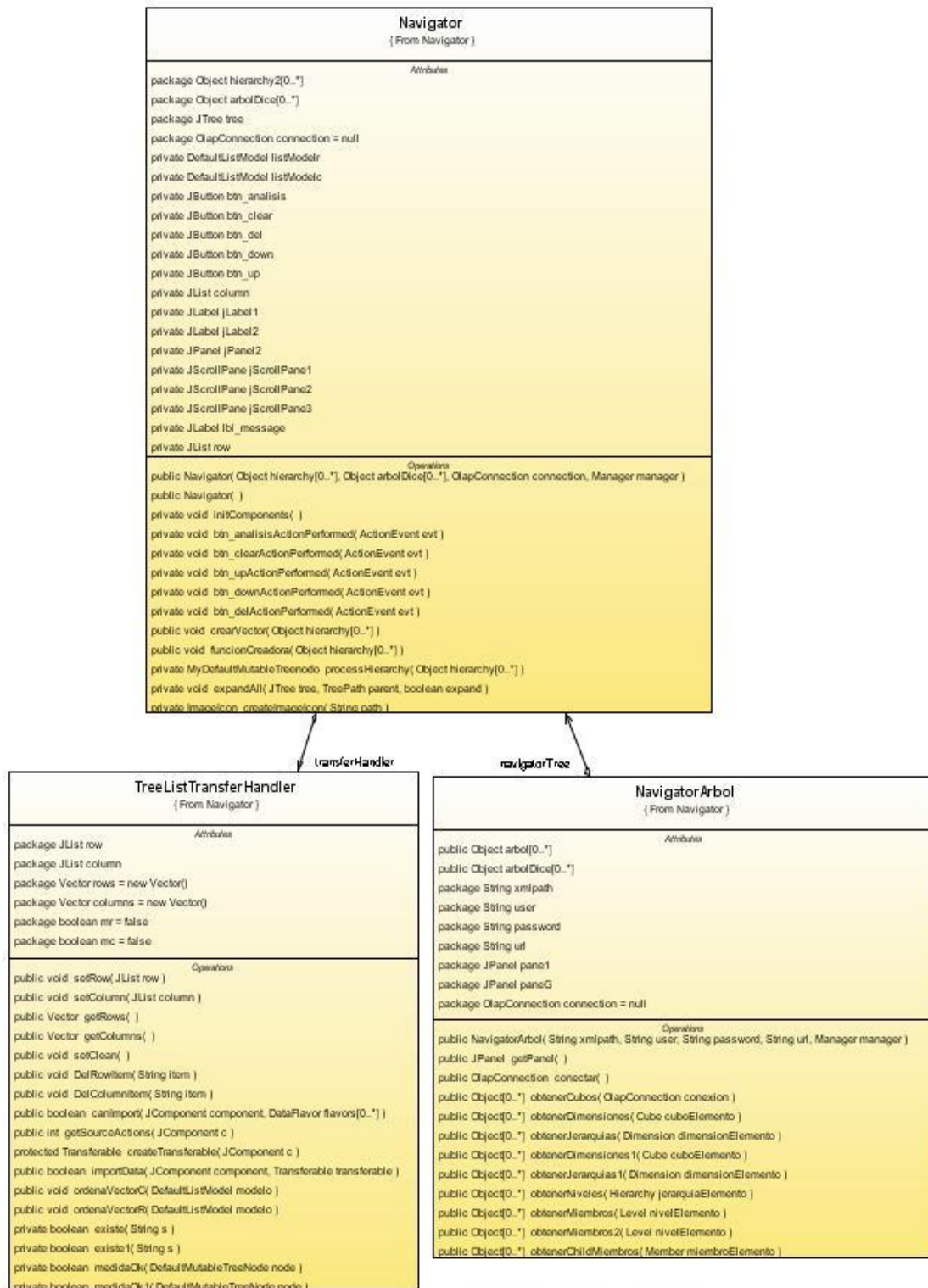


Figura 2.42. Reports

<b>SaveReport</b> { From Reports }
<i>Attributes</i>
<i>Operations</i> public SaveReport( ) private void initComponents(

<b>StarCubeReport</b> { From Reports }
<i>Attributes</i> package int i package JFreeChart chart package JScrollPane table package File file private short ROW_INI = 2 private short COL_INI = 2
<i>Operations</i> public StarCubeReport( int i, JFreeChart chart, JScrollPane table, File file public void runReporte( ) private Image ChartImage( ) private Image TableImage( ) private void exportReport( )

Figura 2.43. Table

TableClap (From Table)
Attributes
<pre>private long serialVersionUID = 1L; private int N_ROWS = 0; private int N_COLS = 0; private int N_HEADER_ROWS = 0; private int N_FIXED_COLS = 0; package Object @quotesFixed[0..*]; package Object @quotesData[0..*]; package String MatrizFixed[0..*]; package String MatrizFixedT[0..*]; package String MatrizdataT[0..*]; package String MatrizdataT[0..*]; package CellSet cellSet; package JScrollPane pane1; package JScrollPane paneG; private int vecTams[0..*] = null; private int vecTamsH[0..*] = null; package int scv = 0; package int sch = 0;</pre>
Operations
<pre>public TableClap( CellSet cellSet, TreeMain tRows, TreeMain tCols, Core core, JScrollPane pane1, JScrollPane paneG ) public void TableClapNew( CellSet cellSet, TreeMain tRows, TreeMain tCols, Core core, JScrollPane pane1, JScrollPane paneG ) private void iniComponentes( ) public SpanTable getFixedTable( ) public SpanTable getTable( ) private SpanTable createDataTable( int headerRow, Object dataT[0..*], Object fixedT[0..*] ) private SpanTable createFixedTable( int headerRow, Object fixedT[0..*], Object fixedT2[0..*] ) private void construiMatrizFixed( CellSet cellSet ) private void construiMatrizdataT( CellSet cellSet ) private void construiMatrizFixedT( CellSet cellSet ) private void construiMatrizFixed2( ) private int span( MyDefaultMutableTreeNode child, SpanModel spanModel, int filainicial ) private int spanH( MyDefaultMutableTreeNode child, SpanModel spanModel, int columnainicial ) private int obtengaindiceReal( int row, int col, Vector vectorMainRow ) private int obtengaindiceReal2( int row, int col, Vector vectorMainCol ) public GClap getGrafico( )</pre>

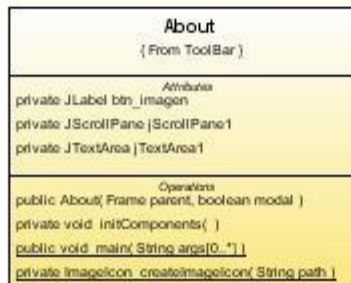
RenderTableCols (From Table)
Attributes
<pre>private Icon iconos[0..*] = null; private Vector vectorMain; private long serialVersionUID = 1L;</pre>
Operations
<pre>public RenderTableCols( TreeMain treeMain ) public Component getTableCellRendererComponent( JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int col ) public boolean validatorChildAncestor( Member member, Member member0 ) private int obtengaindiceReal2( int row, int col, Vector vectorMainCol ) protected void setText( Object value ) protected Style getStyle( SpanTableHeader spanTableHeader, int i, int j ) private ImageIcon createImagenIcon( String path );</pre>

RenderTableRows (From Table)
Attributes
<pre>private Icon iconos[0..*] = null; private Vector vectorMain; private long serialVersionUID = 1L;</pre>
Operations
<pre>public RenderTableRows( TreeMain treeMain ) public Component getTableCellRendererComponent( JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int col ) public boolean validatorChildAncestor( Member member, Member member0 ) private int obtengaindiceReal( int row, int col, Vector vectorMainRow ) protected void setText( Object value ) protected Style getStyle( SpanTable spanTable, int i, int j ) private ImageIcon createImagenIcon( String path );</pre>

HeaderTableModel1 (From Table)
Attributes
<pre>private long serialVersionUID = 1L; private int nRow; private TableColumnModel cm; package Object fixedT[0..*];</pre>
Operations
<pre>package HeaderTableModel1( int headerRow, TableColumnModel columnModel, Object fixedT[0..*] ) public Object getValueAt( int row, int col ) public int getRowCount( ) public int getColumnCount( )</pre>



Figura 2.44. Toolbar



## 5. IMPLEMENTACIÓN

La implementación de la herramienta STARCUBE, se realizó sobre:

- **Sistema operativo:** Microsoft® Windows® XP SP2.
- **Lenguaje de programación:** Java™ 6.0.
- **Plataforma de desarrollo:** NetBeans™ 6.0, como IDE de desarrollo.

### 5.1 ARQUITECTURA

La arquitectura general de la herramienta STARCUBE se muestra en la figura 2.45 en la cual se visualiza la interacción de esta herramienta con la API OLAP4J y el SGBD PostgreSQL.

En la figura 2.46 se presenta la arquitectura interna de STARCUBE, la cual está conformada por tres módulos:

**Módulo de utilidades:** Este módulo tiene 2 tareas principales:

- Realizar la conexión a la base de datos.
- Contener la colección de clases principales y bibliotecas que son utilizadas por otras clases en la manipulación y visualización de los datos, de esta manera puede hacerse la administración de las mismas y se hace factible la reutilización de código, una de estas bibliotecas corresponde a la API “OLAP4j driver for Mondrian”, la encargada de ejecutar las consultas en lenguaje MDX.

**Módulo de kernel:** En este módulo se encuentran los módulos de la lógica de OLAP, necesario para poder crear y manipular cubos y realizar un correcto análisis de los datos y así poder tomar las decisiones más acertadas.

En la parte de creación de cubos se realiza la escogencia de tablas a hacer uso de la Data Warehouse, además de las medidas de la tabla de hechos y el diseño de las dimensiones del cubo con sus respectivas jerarquías. Ya en la manipulación de cubos, se arma el análisis que desee el usuario y “juega” con ese análisis, manipulándolo de la manera que desee, aplicándole operaciones como Dice, Slice, Drill Down, Roll Up y Pivot, además de manejar la medidas y observar sólo las que quiera ver, se incluye también de la eliminación de elementos de la tabla que el usuario no necesite.

Está compuesto de 2 submódulos principales:

- **Módulo de creación de cubos:** En este módulo se realizan las operaciones necesarias para poder crear los cubos, con las dimensiones y medidas que el usuario escoja.
- **Módulo de manipulación de cubos:** Este módulo es el encargado de crear el análisis y soportar todas las operaciones que se le pueden realizar sobre el mismo, además tiene la tarea importantísima de mantener la consistencia de la información que se le está presentando al usuario y también de conservar la interactividad con el mismo.

**Módulo de GUI (Interfaz Gráfica de Usuario):** Es el módulo encargado de realizar la interfaz entre el usuario y la lógica de la herramienta, ocultando la complejidad de los procesos y suministrándole al usuario un ambiente agradable de trabajo y muy intuitivo para que interactúe de la manera que el más se adecúe.

Figura 2.45. Arquitectura general de STARCUBE

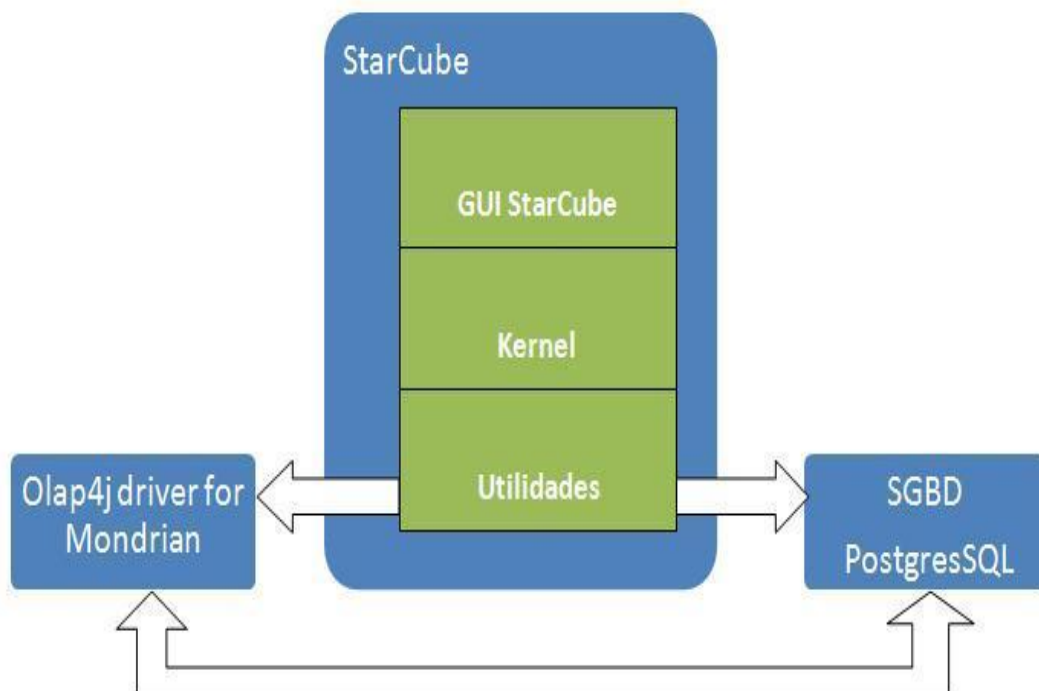
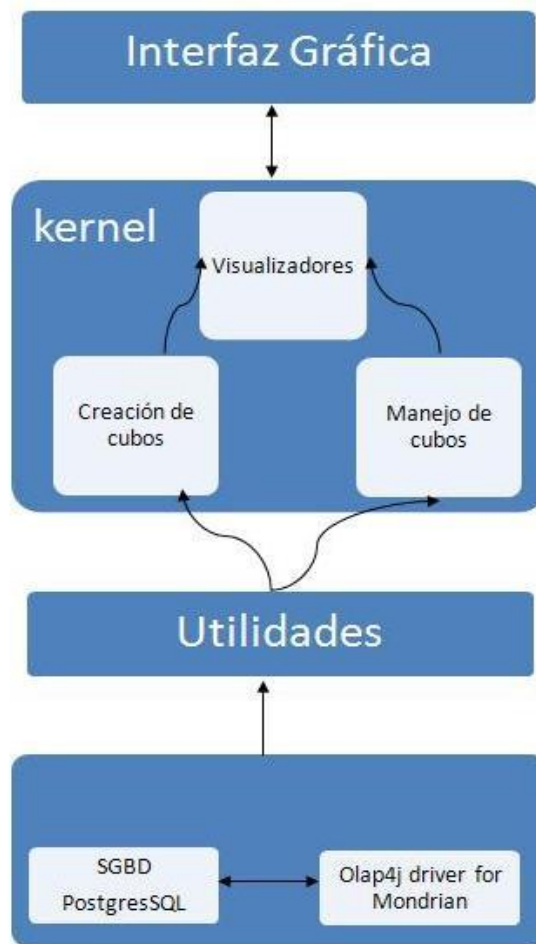


Figura 2.46. Arquitectura interna de STARCUBE



## 5.2 ESTRUCTURA DE PAQUETES DE LA APLICACIÓN

La estructura general de los paquetes de la herramienta STARCUBE es la siguiente:

- ✓ Módulo de kernel. El módulo de kernel está compuesto por los siguientes paquetes:
  - Paquete CubeConfiguration. Contiene las clases necesarias para la creación de cubos incluyendo la selección de medidas y de dimensiones, también clases encargadas de generar el archivo XML correspondiente al archivo de configuración del cubo y también de almacenarlo.
  - Paquete Main. Contiene la clase que es el núcleo del manejador de cubos.
  - Paquete QueryMdx. Contiene las clases necesarias para el manejo de la consulta MDX.

- Paquete operaciones. Contiene las clases necesarias para soportar el uso de operaciones del análisis del cubo, tales como dice, Drill Down, Roll Up, Pivot, y el manejo de medidas.
- ✓ Módulo GUI. Contiene el paquete WorkSpace.
  - Paquete WorkSpace. Contiene las clases utilizadas en el espacio de trabajo de la herramienta Software STARCUBE.
- ✓ Módulo Utils. En él se implementan los paquetes y las clases para extraer los datos de la base de datos, también las clases necesarias para poder ejecutar las consultas en lenguaje MDX, entre las que se encuentran las bibliotecas de clases de la API OLAP4J y también se encuentran todas las clases que son utilizadas por las demás clases del proyecto.

A continuación se amplía y se detallan conceptos sobre los paquetes más importantes dentro del desarrollo de la herramienta y la manera como fueron implementadas las mismas.

#### 5.2.1 Módulo de kernel

##### 5.2.1.1 Paquete cubeconfiguration.

Las clases principales de este paquete son CubeSchemaWriter.java y FileStorage.java, la primera es la encargada de crear la configuración de cubo mediante la utilización de etiquetas xml, como resultado se tiene un archivo xml con toda la información seleccionada por el usuario en el proceso de creación del cubo, la segunda clase cumple la función de leer y guardar el archivo xml en la base de datos.

##### 5.2.1.2 Paquete main.

Este paquete está compuesto por la clase Core.

**Clase Core:** La clase Core es la más importante en lo que a manipulación de análisis se refiere. Esta clase recibe el análisis de la clase NavigatorArbol perteneciente al subpaquete Navigator del paquete WorkSpace. Dicho análisis se compone de un Vector de elementos escogidos por el usuario por cada eje de la tabla que él quiere visualizar. Un objeto de tipo Core contiene dos objetos que son instancias de las clases Cols y Rows respectivamente, pertenecientes al paquete QueryMDX, durante todo el ciclo de vida del objeto. Cada vector será enviado a cada uno de los objetos en mención, para que ellos puedan instanciarse. Con estos objetos, cols y rows, el objeto de la clase Core pretende construir la consulta MDX, ejecutarla a través de algunas de las clases pertenecientes al paquete Utils, obtener los resultados de dicha consulta y entregárselos a una instancia de la Clase Manager perteneciente al subpaquete ManagerCube del paquete Core.

La clase Core es la encargada de administrar las operaciones que se realizan al análisis, es decir que cada una de las operaciones que se pueden realizar en OLAP pasan primero por la instancia de esta Clase quien analiza que operación es la requerida por el usuario y la envía a los objetos correspondientes. Estos objetos devuelven los resultados, e inmediatamente se tienen que actualizar todos los atributos de este objeto.

Los principales métodos de esta clase, son:

- createMembers(): le entrega los vectores de jerarquías escogidos por los usuarios a Cols y Rows.
- dice(): es el encargado de manejar todo lo que tenga que ver con la operación dice.
- pivot: intercambia los ejes, los atributos de la instancia de la clase Rows pasan a ser atributos de la instancia de la clase Cols, y viceversa.
- drillDown\_RollUp(): maneja las operaciones de navegación por la estructura jerárquica, Drill Down y Roll Up.
- deleteMember(): elimina elementos del análisis a la escogencia del usuario.
- addMeasureTable(), deleteMeasureTable(): métodos encargados del manejo de las medidas.

#### 5.2.1.3 Paquete queryMDX.

En este paquete se destacan las siguientes clases:

**Clase Cols:** encargada de manejar todo lo concerniente al eje de las columnas de la tabla. Recibe como constructor el vector de jerarquías proporcionado por la clase Core, con las cuales construye un árbol, que contiene todos los elementos correspondientes a las jerarquías, durante toda la existencia del Core. Con este árbol se construye una parte de la consulta MDX, además en éste árbol se manipulan a lo largo de su existencia todos los elementos del análisis y sobre él recaen todas las operaciones.

Los principales métodos de esta clase, son:

- drillDown\_RollUp(): método encargado de soportar las operaciones de navegación por la estructura jerárquica del árbol que tiene asociado.
- addMeasure(): añade una medida al árbol que tiene.
- deleteMeasure: elimina la medida que el usuario escogió del eje de las columnas.
- deleteMember(): elimina un elemento del árbol y por ende del eje de las columnas.
- getVectorMain(): retorna el árbol que está asociado a Cols, para que se le realice alguna de las operaciones.
- setVectorMain(): intercambia el árbol que posee por otro que viene desde afuera de esta clase, resultado de una operación.

**Clase Rows:** es la encargada de manejar todo lo que tiene que ver con el eje de las filas. Al igual que la clase Cols, recibe como constructor el vector de jerarquías proporcionado por la clase Core, con las cuales construye un árbol, que contiene todos los elementos correspondientes a las jerarquías, durante toda la existencia del Core. Con este árbol se construye otra parte de la consulta MDX, además en éste árbol se manipulan a lo largo de su existencia todos los elementos del análisis y sobre él recaen todas las operaciones.

Los métodos de la clase Rows son los mismos que los métodos de la clase Cols. La diferencia entre estas 2 clases es que cada una tiene un árbol diferente asociado al de la otra clase.

**Clase QueryAxis:** Es la clase encargada de recorrer el árbol asociado a cada uno de los ejes (instancias de las clases Rows y Cols respectivamente) y construir una parte de la consulta MDX con cada eje.

El método más importante de esta clase es:

- `createQuery():` es el método principal de esta clase ya que llamando a otros submétodos recorre de manera recursiva un árbol a la vez, y construye la parte de la consulta asociada al árbol.

**Clase Query:** Es la clase encargada de armar todas las partes de la consulta MDX, lo concerniente a la clase Rows, Cols y la respectiva restricción (Dice) en caso de que exista.

El método más importante es:

- `createQuery():` quien recibe como parámetro la parte de la consulta MDX de Cols, de Rows y del Dice y arma una sola consulta, una instancia de la clase String, para posteriormente ser ejecutada.

**Clase ExecuteQuery:** Es la clase encargada de recibir la consulta que arma la instancia de la clase Query y la ejecuta a través de algunas de las clases pertenecientes al paquete Utils.

Su método más importante es:

- `executeQuery():` quien es el encargado de ejecutar la consulta y almacenar los datos obtenidos tras la ejecución en uno de sus atributos para que posteriormente puedan ser retornados hacia la clase Manager perteneciente al subpaquete ManagerCube del paquete Core.

#### 5.2.1.4 Paquete operations.

De éste paquete se destacan las siguientes clases:

**Clase Dice:** Es la encargada de ponerle una restricción al análisis sin la necesidad de tener que anexar toda la jerarquía de elementos en el análisis, sólo toma el elemento que el usuario ha escogido para luego adicionarla de restricción al análisis.

Sus métodos más importantes son:

- `crearqueryDice()`: aquí se crea la última sección de la consulta MDX, donde va la restricción con los elementos seleccionados, esta sección de la consulta será retornada a la instancia de la clase Query del subpaquete `queryMDX` para que termine de armar la consulta MDX.
- `setElementoDice()`: quien es el encargado de adicionar algún elemento que al usuario se le ocurra para realizar una consulta mucho más restringida acercada a lo que él necesita.

**Clase DrillDown\_RollUp:** Es la clase encargada de realizar las operaciones de navegación por la estructura jerárquica de una de las instancias de Cols o Rows, y determinar si lo que el usuario quiere realizar Drill Down o Roll Up y enviar a cada una de estas clases lo necesario para ejecutar la operación correspondiente.

El método más importante de esta clase, es:

- `verificate()`: método encargado de revisar que es lo que el usuario realmente quiere, Drill Down o Roll Up y de enviar a la Clase DrillDown o Roll Up el árbol correspondiente al eje a modificar, para que le realice la operación necesaria.

**Clase Drill Down:** Es la clase que realiza la operación de Drill Down al árbol del eje implicado. Esta clase se ve ayudada por algunas de las clases del paquete Utils.

El método más importante de esta clase, es:

- `operations()`: quien es el encargado de realizar la operación de Drill Down directamente al árbol y de reconstruirlo totalmente, garantizando la consistencia de sus niveles.

**Clase Roll Up:** Realiza la operación Roll Up al árbol que le entreguen. Esta clase se ve ayudada por algunas de las clases del paquete Utils.

El método más destacado de esta clase, es:

- `operations()`: quien realiza la operación de Roll Up al árbol y lo reconstruirlo totalmente, garantizando la consistencia de sus niveles.

#### 5.2.1.5 Paquete managercube.

Contiene la siguiente clase:

**Clase Manager:** Es la encargada de realizar la interacción entre la instancia de la clase Core del subpaquete main del paquete Core y las instancias de las clases encargadas de la visualización del cubo o desde su perspectiva, del análisis; en otras palabras es quien hace el puente entre la interfaz de Usuario y la instancia de Core. Esta clase maneja la tabla, la gráfica y el Core. El extrae los datos de la consulta realizada por el Core, y se los entrega a la tabla y a la gráfica para que puedan ser visualizados, además interviene en la realización de peticiones de



operaciones sobre el análisis al Core, que envíe el usuario.

Su método más importante, es:

- `printAll()`: es el método que más se destaca en esta clase. Contiene rutinas para enviar al espacio de trabajo los datos necesarios para ser visualizados por el usuario.

## 5.2.2 Módulo GUI

### 5.2.2.1 Paquete workspace.

La interfaz gráfica de la aplicación se desarrolló mediante la utilización del proyecto “MyDoggy”, es un framework (marco de trabajo) de acoplamiento, para la gestión de ventanas secundarias dentro de una ventana principal, además de esto, se trabaja utilizando las funcionalidades del proyecto *Matisse*, el constructor de interfaces gráficas de usuario (GUI) propia de *NetBeans 6.0* que proporciona un diseño visual de las formas y su posterior programación.

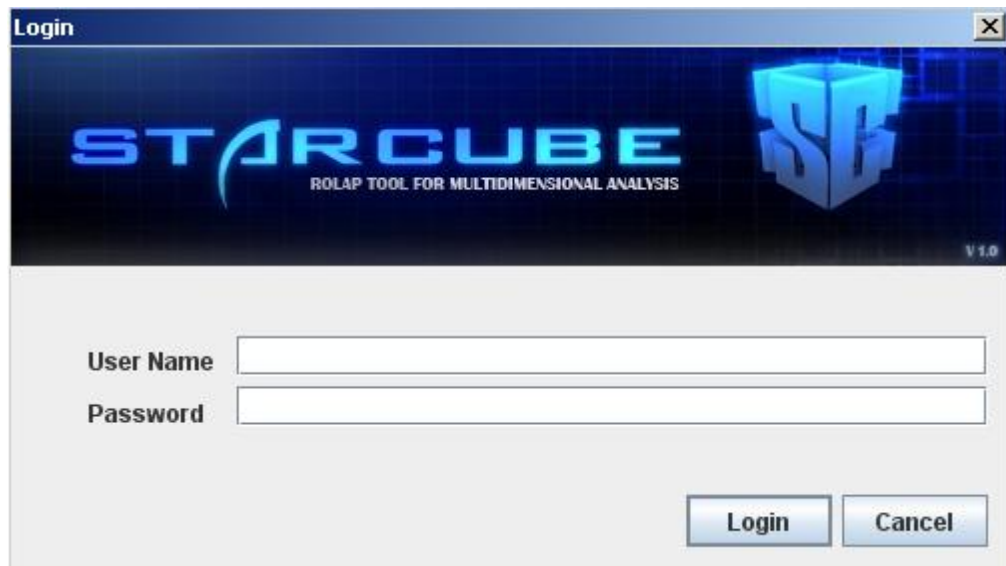
El paquete *Workspace* contiene los siguientes paquetes:

- *Main*, tiene todo lo referente a las configuraciones iniciales de la aplicación.
- *Account*, contiene los archivos para la administración de usuarios.
- *Cube*, presenta los cubos creados por el usuario y los compartidos por otros.
- *CubeConfiguration*, permite configurar y crear un cubo.
- *DBconexion*, contiene los archivos para establecer una conexión a una fuente de datos.
- *Dice*, maneja las restricciones del cubo.
- *Graphic*, se encarga de la etapa de visualización de los datos del análisis actual, de forma gráfica.
- *Help*, permite mostrar la ayuda.
- *Measures*, maneja las medidas del cubo.
- *Navigator*, configura el análisis inicial del cubo y modificaciones posteriores.
- *Reports*, contiene los archivos para generar reportes del análisis actual.
- *Table*, se encarga de la etapa de visualización de los datos del análisis actual, en una tabla.

#### 5.2.2.1.1 Main.

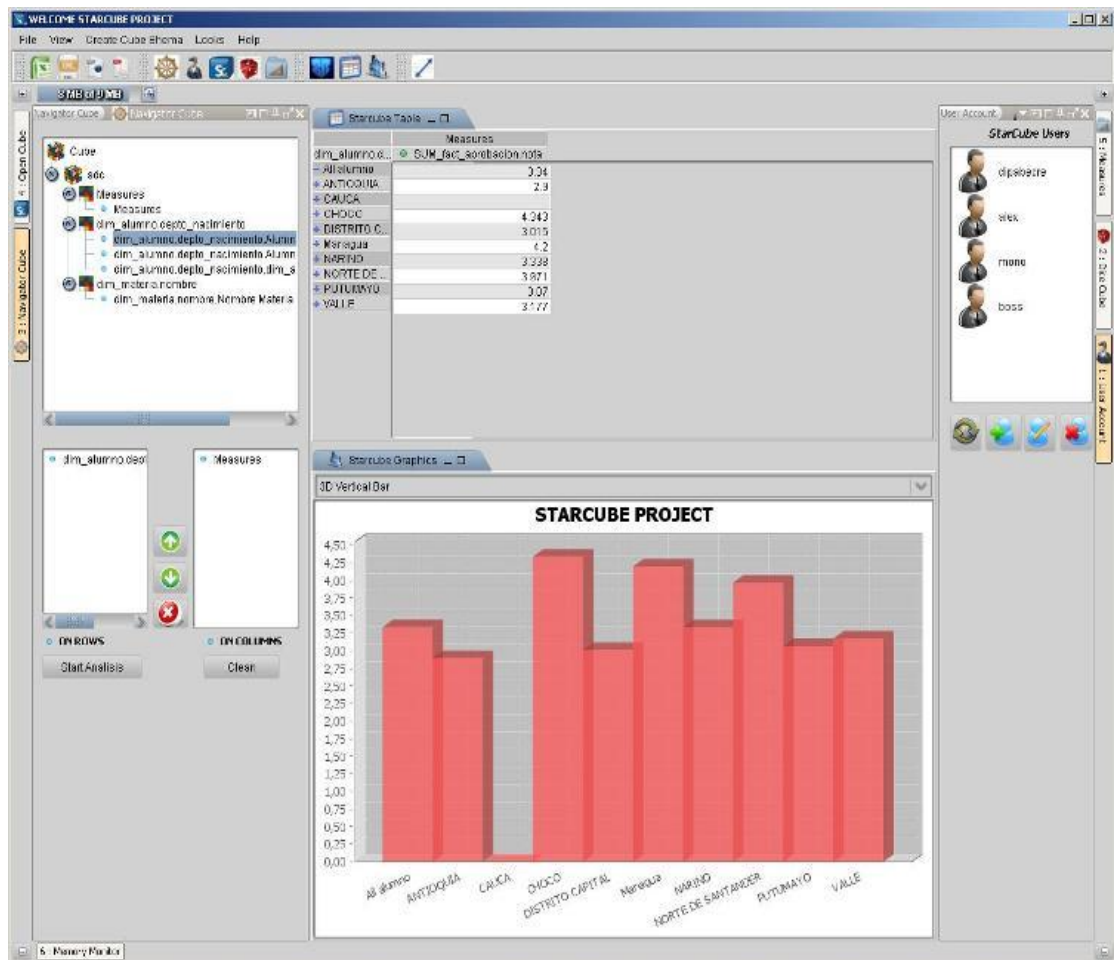
La clase principal de este paquete es STARCUBEDoggySet.java, esta clase permite realizar la autenticación de usuarios, mediante la utilización de unos componentes, del proyecto “SwingX” el cual desarrolla extensiones de los componentes de Swing de java, que permite generar el formulario de autenticación, en la figura 2.47 se aprecia el formulario de autenticación de usuarios.

Figura 2.47. Autenticación de usuarios



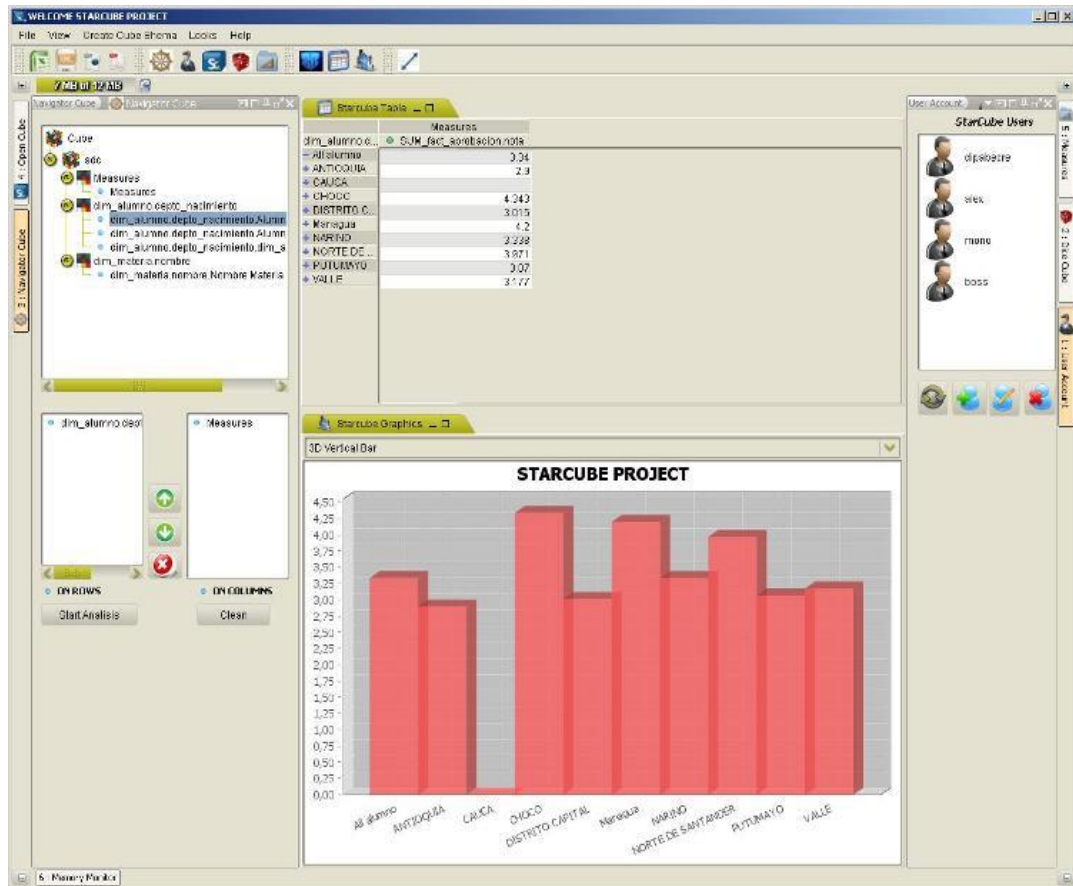
Además ésta utiliza algunos componentes del proyecto “MyDoggy”, componentes como **ToolWindow**, **ContentManager**, que permiten crear el marco de trabajo de la aplicación que se observa en la figura 2.48.

Figura 2.48. Marco de trabajo



Aparte de lo anterior se realiza la carga de la información, del usuario que se autentica en la aplicación, como son el nombre de usuario, la apariencia de la aplicación y el tipo de usuario, en la figura 2.49, se presenta una de las apariencias que puede tener la aplicación.

Figura 2.49. Apariencia marco de trabajo



5.2.2.1.2 Account. Contiene los siguientes formularios:

Account.java. Es el formulario que presenta los usuarios del sistema (Figura 2.50).

Figura 2.50. Usuarios del sistema.



NewAccount.java. Formulario para la creación y edición de usuarios de la aplicación. En las figuras 2.51 y 2.52 se observa este formulario.

Figura 2.51. Edición de usuario

User Account Account

**Role Information**

Rol Name

Password

Super User  (yes or not)

**Personal Information**

Name

Last Name

**Look And Feel**

Ch...

Create User

5 : Measures

2 : Dice Cube

1 : User Account

Figura 2.52. Creación de usuario.

User Account Account

**Role Information**

Rol Name

Password

Super User  (yes or not)

**Personal Information**

Name

Last Name

**Look And Feel**

Steel

Create User

5 : Measures

2 : Dice Cube

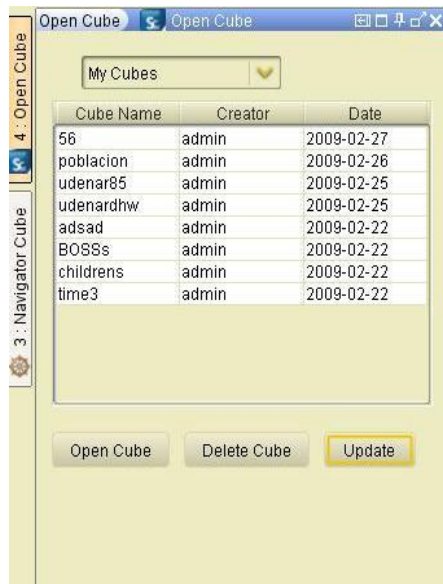
1 : User Account

### 5.2.2.1.3 Cube.

MyCubes.java. Formulario que presenta el listado de cubos, utilizando un filtro

(lista desplegable de cubos privados o públicos). En la figuras 2.53 y 2.54 se ve este formulario.

Figura 2.53. Cubos creados.



El segundo lista los cubos compartidos por los demás usuarios del sistema.

Figura 2.54. Cubos compartidos.



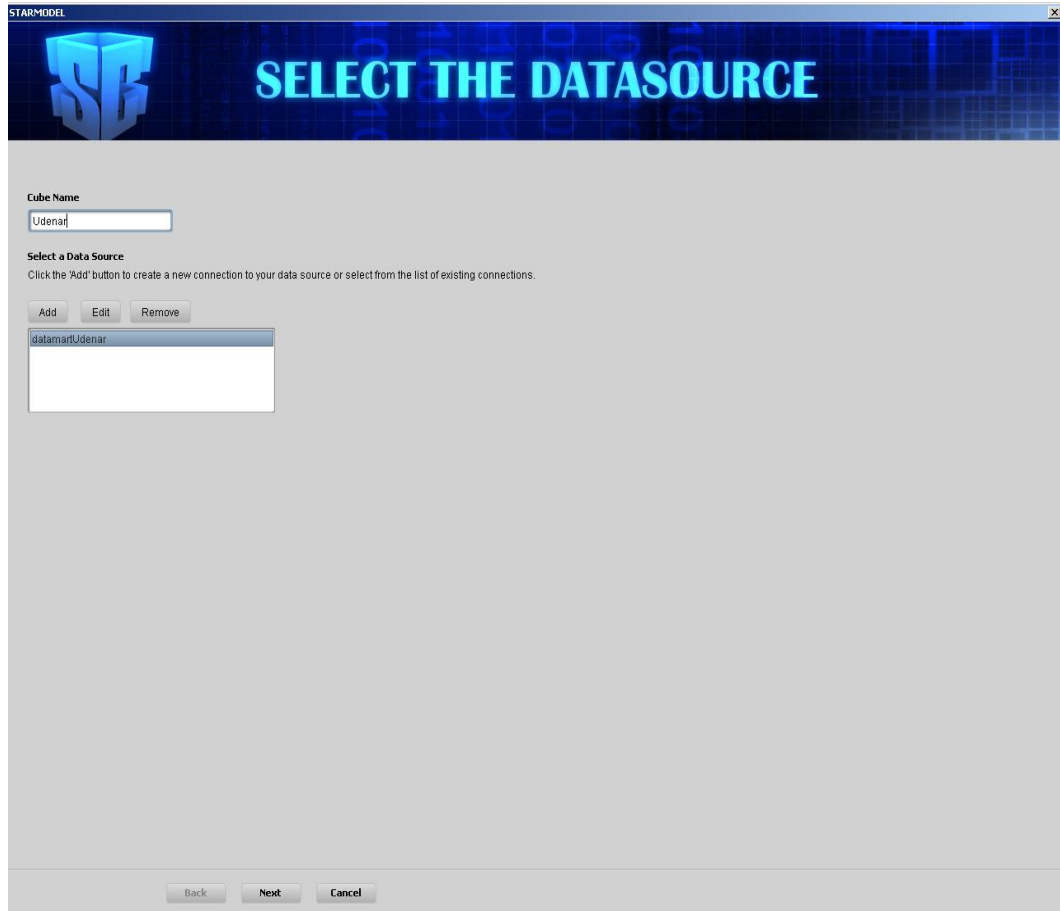
#### 5.2.2.1.4 Cubeconfiguration.

La clase principal de este paquete es Wizard\_ConfCube.java, la cual presenta un asistente en la creación de los cubos.

El primer formulario del asistente permite, crear una conexión a una fuente de datos además de ponerle un nombre al cubo que se va a crear (Figura 2.55).

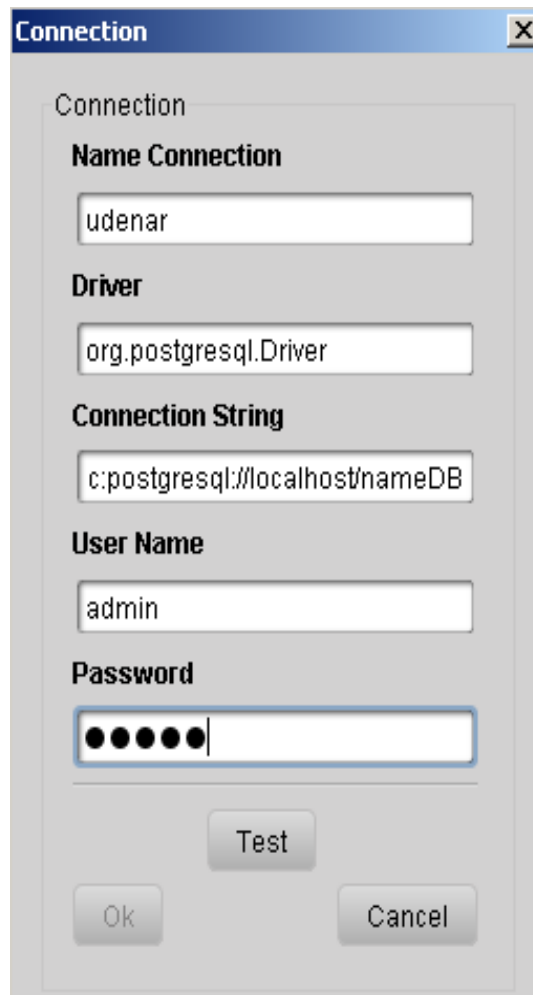


Figura 2.55. Nombre del cubo.



El segundo formulario del asistente presenta las relaciones de la fuente de datos, las cuales se pueden seleccionar o arrastrar y soltar en el área blanca del formulario (Figura 2.56).

Figura 2.56. Configuración fuente de datos

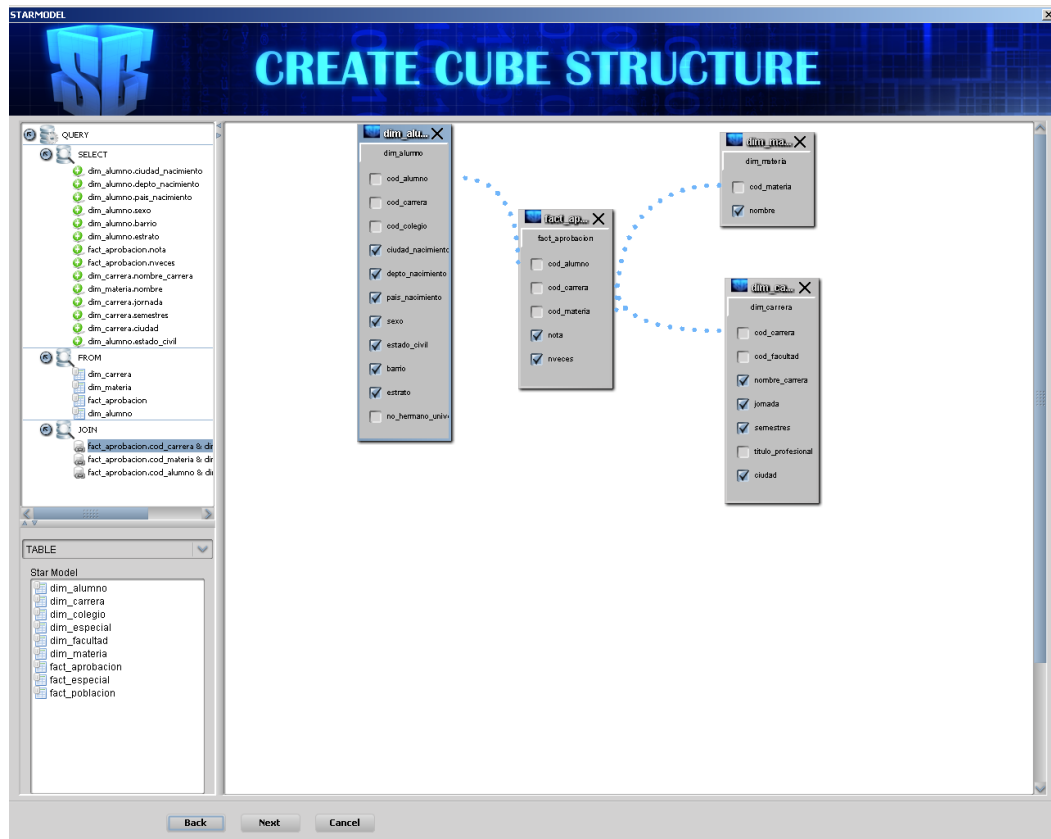


The image shows a 'Connection' dialog box with the following fields and controls:

- Connection** (Section Header)
- Name Connection**: Text input field containing 'udenar'.
- Driver**: Text input field containing 'org.postgresql.Driver'.
- Connection String**: Text input field containing 'c:postgresql://localhost/nameDB'.
- User Name**: Text input field containing 'admin'.
- Password**: Password input field with five black dots and a cursor.
- Test**: Button.
- Ok**: Button.
- Cancel**: Button.

Se puede seleccionar campos de las relaciones, ubicadas en el espacio de trabajo además de realizar conexiones entre campos arrastrando y soltando las etiquetas de los mismos (Figura 2.57).

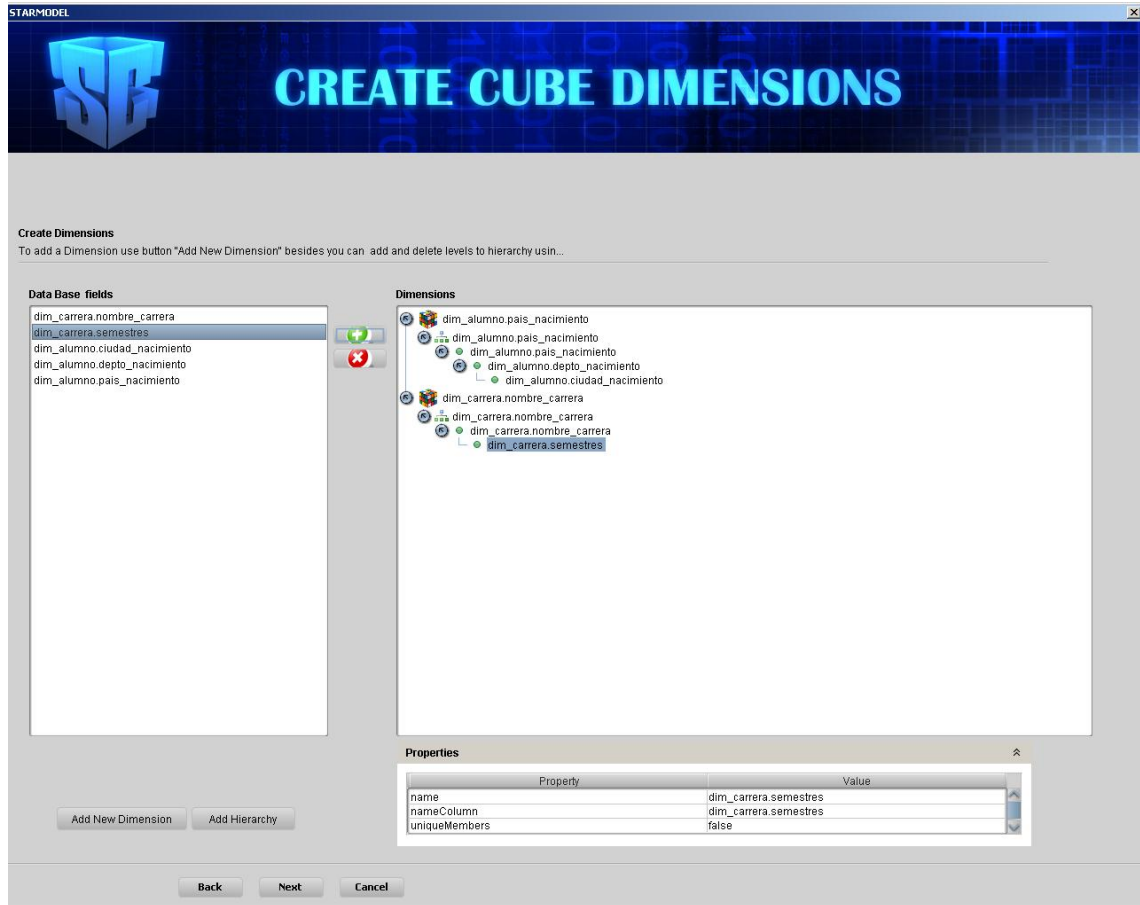
Figura 2.57. Conexiones y selección de campos.



El tercer formulario lista los campos seleccionados en el paso anterior, todo esto para escoger que campos serán las medias del cubo (Figura 2.58).



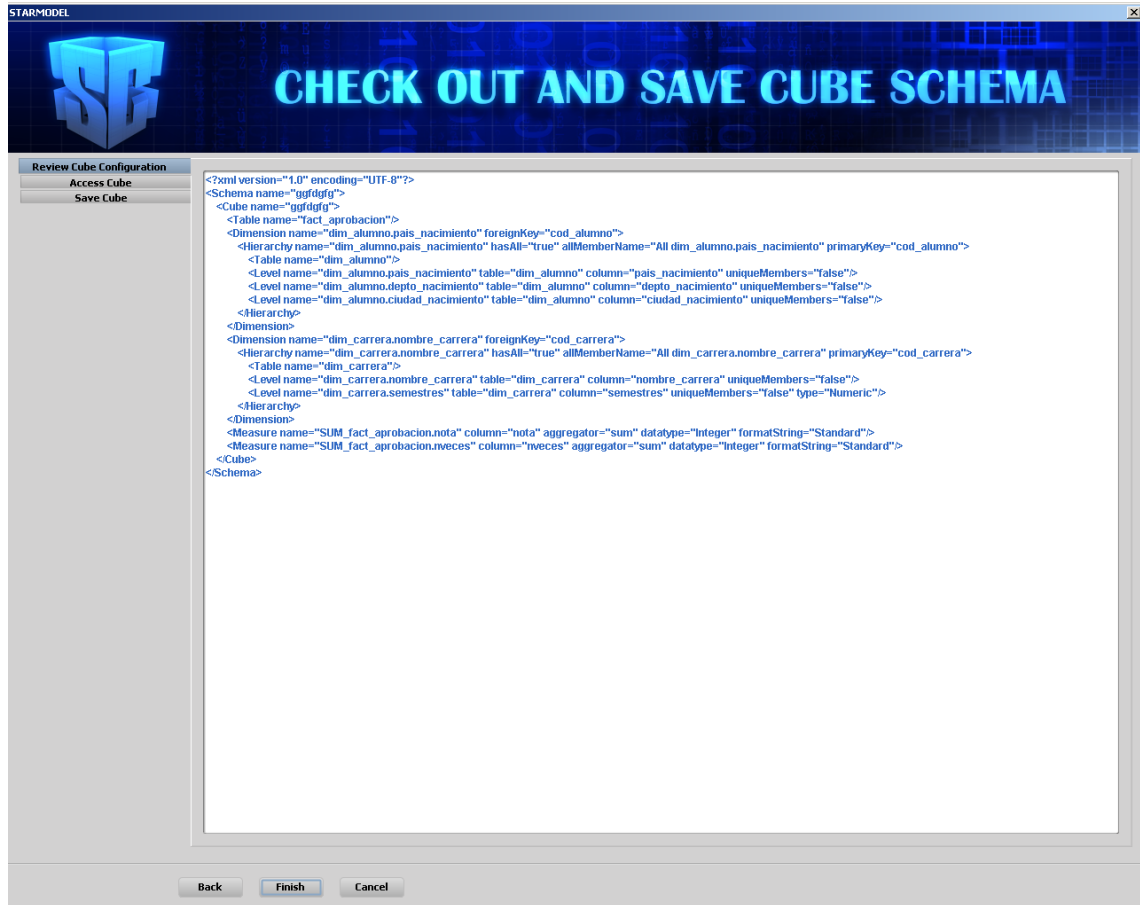
Figura 2.59. Dimensiones del cubo.



El quinto formulario contiene tres subformularios:

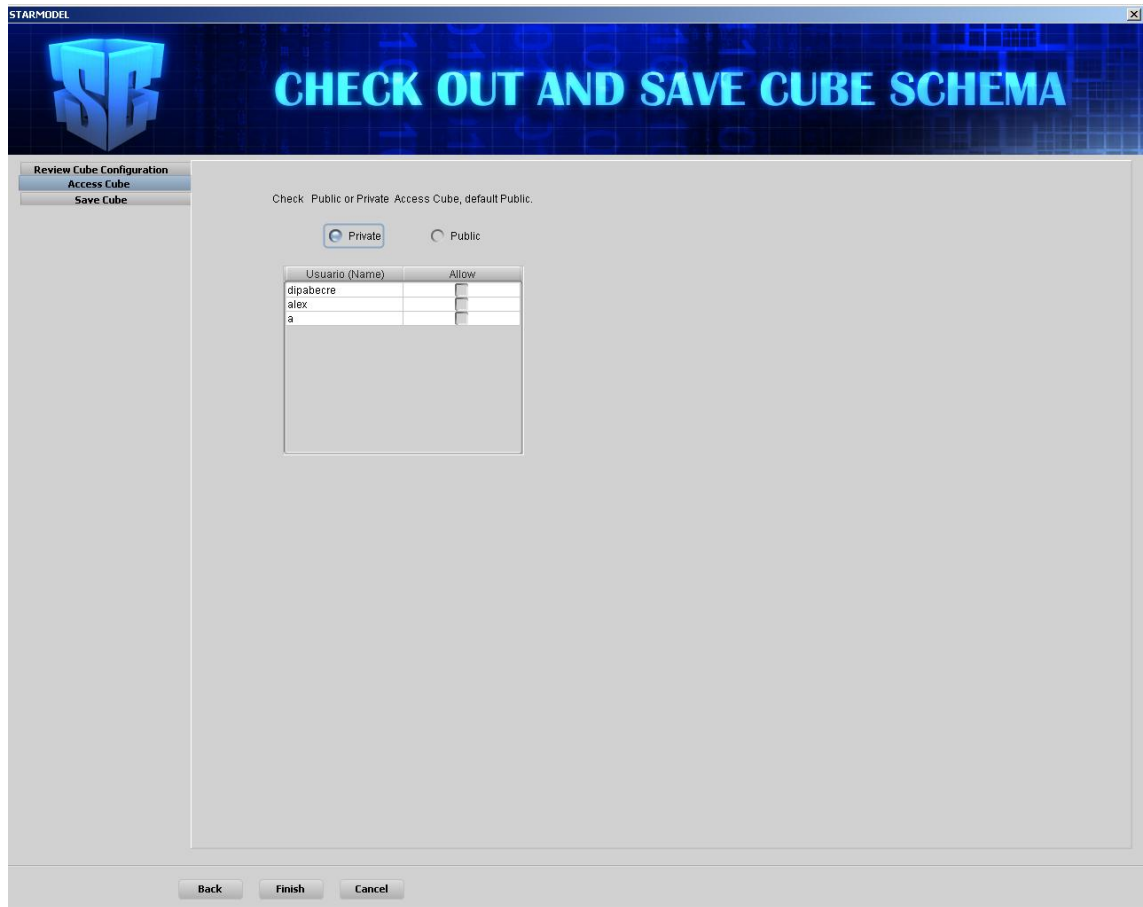
El primero presenta toda la configuración realizada sobre el cubo, lo anterior será realizado si realiza bien su definición (Figura 2.60).

Figura 2.60. Revisión del cubo.



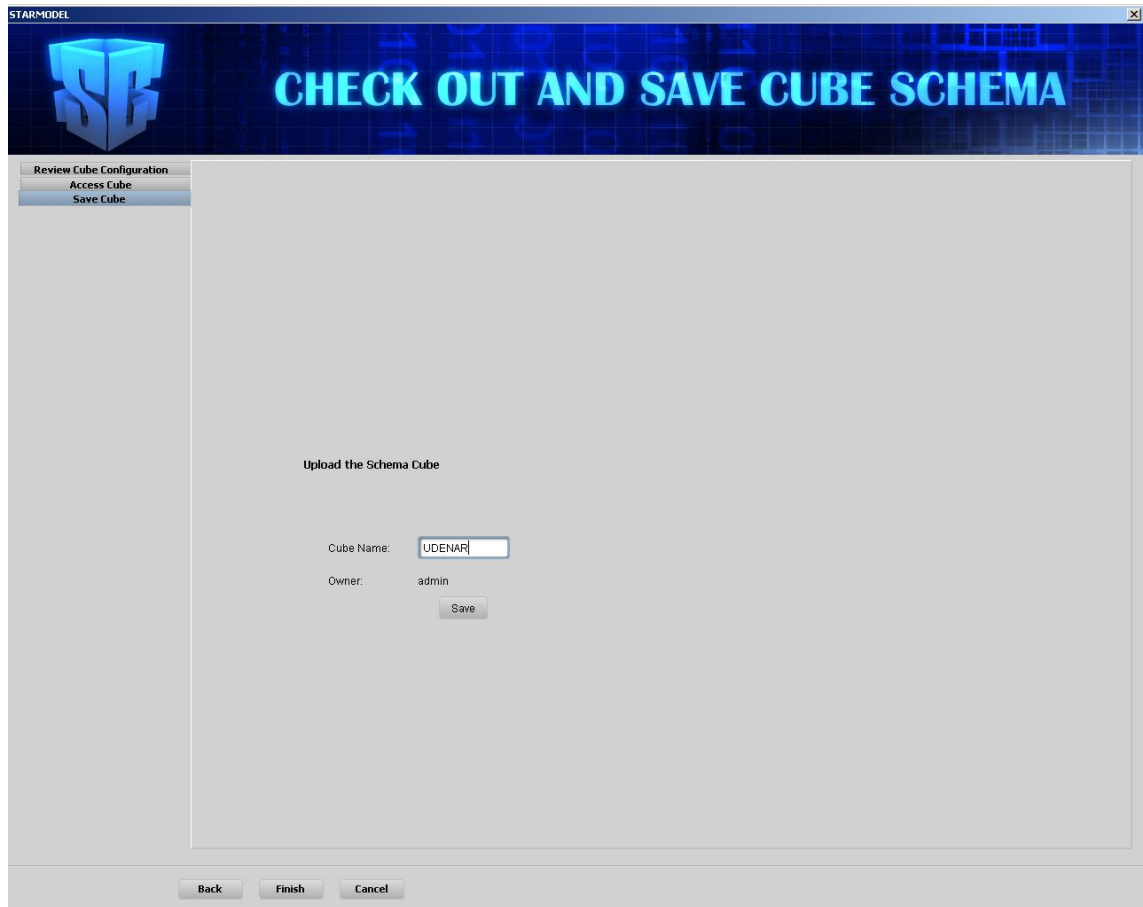
El segundo permite escoger los permisos de acceso al cubo (Figura 2.61).

Figura 2.61. Permisos del cubo.



El tercero guarda el cubo de ser exitosa la definición del mismo (Figura 2.62).

Figura 2.62. Guardar el cubo.



#### 5.2.2.1.5 Graphic.

GOLAP.java. Es la clase principal de este paquete, es el encargado de generar diferentes tipos de gráficos, luego de iniciar un análisis en la aplicación (Figuras 2.63 y 2.64).



Figura 2.63. Gráfico de barras apiladas

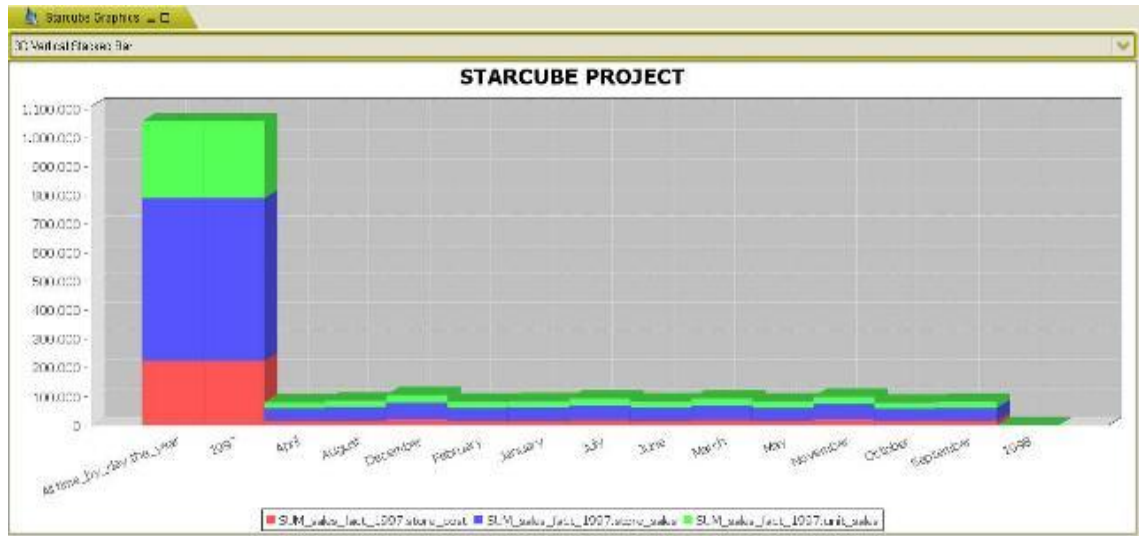
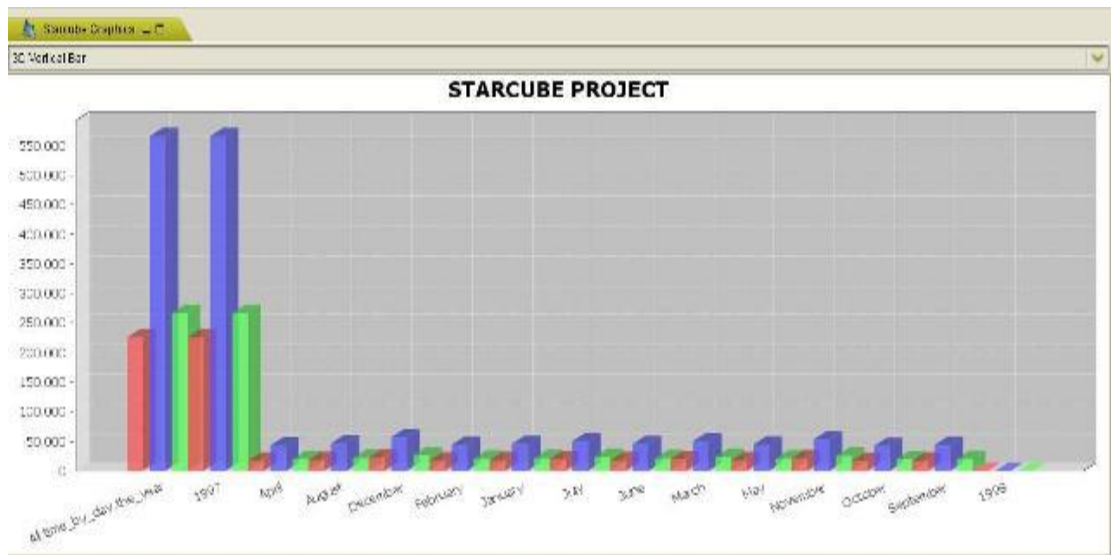


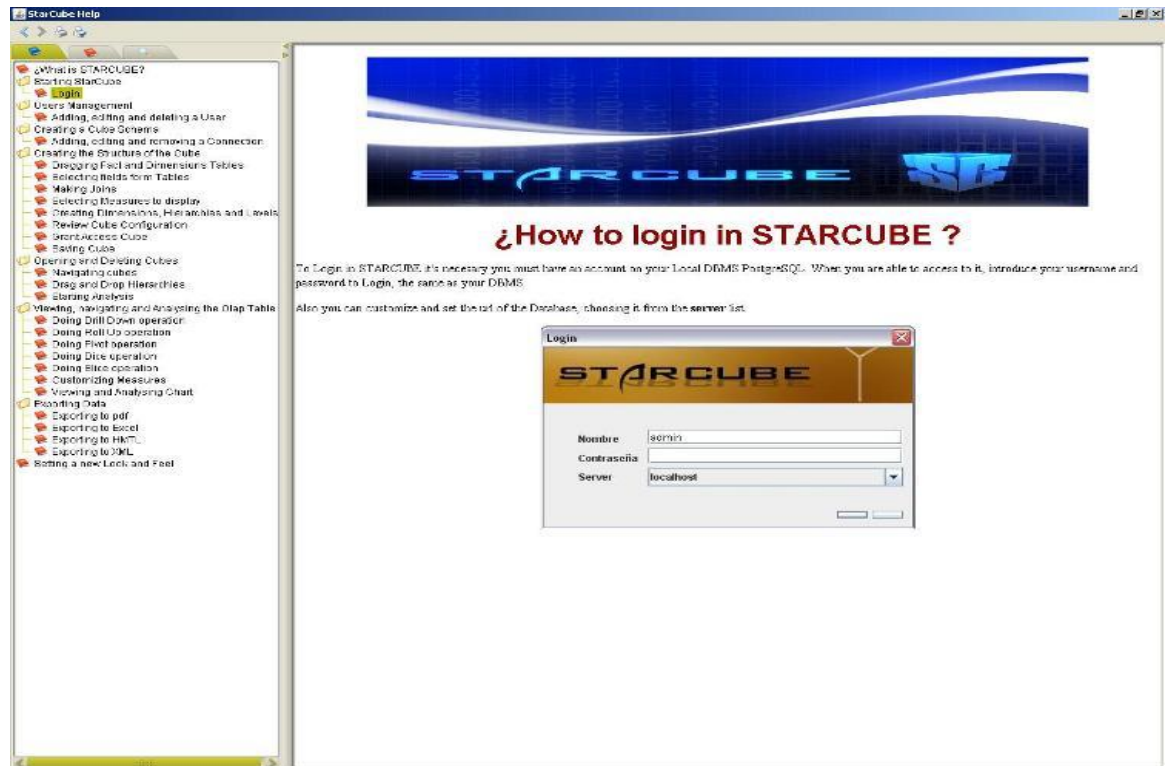
Figura 2.64. Gráfico de barras



### 5.2.2.1.6 Help.

Mediante la utilización del proyecto “javahelp”, fue posible generar la ayuda de usuario, la cual presenta el manejo de la globalidad de la aplicación (Figura 2.65).

Figura 2.65. Ayuda de usuario.



### 5.2.2.1.7 Measures.

Formulario para el manejo de las medidas del cubo (Figura 2.66).

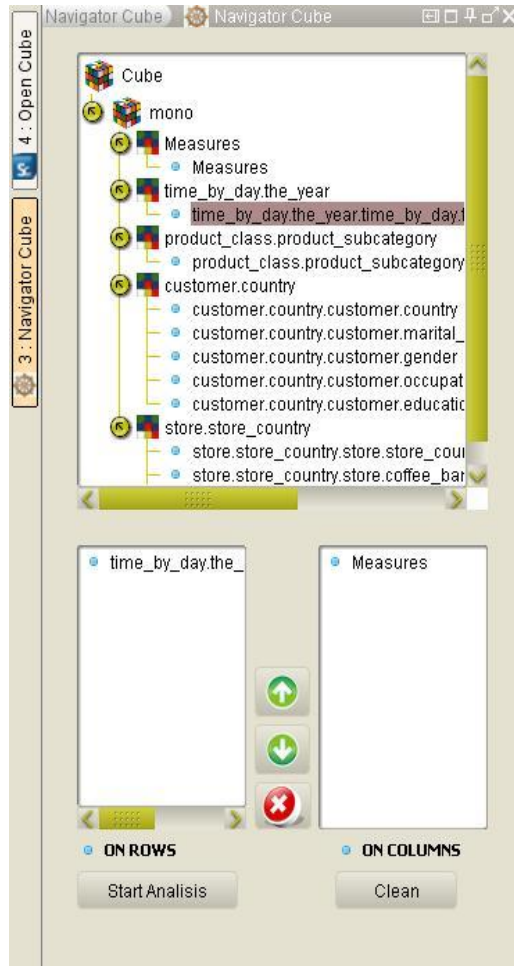
Figura 2.66. Medidas del cubo.



#### 5.2.2.1.8 Navigator.

La clase principal de este paquete es Navigator.java, configura el análisis inicial del cubo y modificaciones posteriores (Figura 2.67).

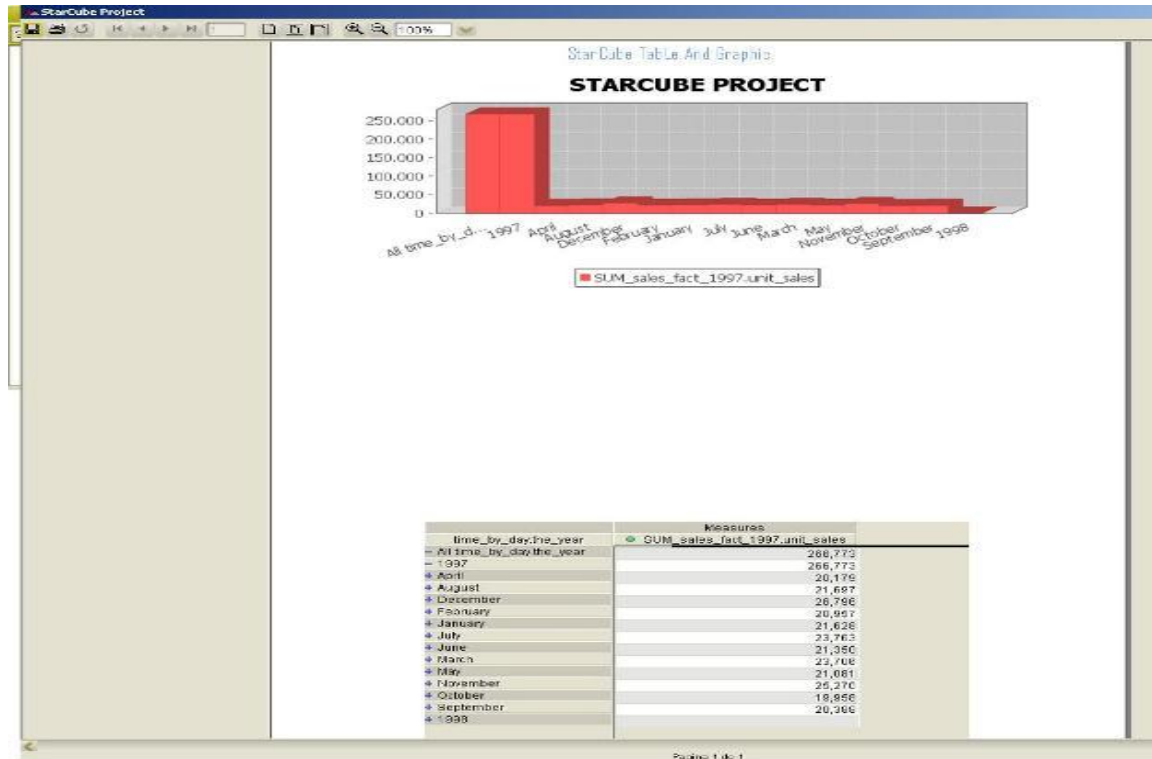
Figura 2.67. Navegador del cubo.



#### 5.2.2.1.9 Reports.

Formulario para guardar el reporte generado del análisis actual (Figura 2.68).

Figura 2.68. Reporte de un análisis



### 5.2.2.1.10 Table.

La clase principal de este paquete es TableOLAP.java, la cual se encarga de la etapa de visualización de los datos del análisis actual, en una tabla (Figura 2.69).

Figura 2.69. Datos del análisis.

	Measures	Measures	Measures
time_by_day.the_year	SUM_sales_fact_1997.store_cost	SUM_sales_fact_1997.store_sales	SUM_sales_fact_1997.unit_sales
All time_by_day.the_year	225,627	565,238	266,773
1997	225,627	565,238	266,773
April	17,112	42,878	20,179
August	18,436	46,199	21,697
December	22,746	56,966	26,796
February	17,600	44,059	20,957
January	18,178	45,540	21,628
July	20,006	50,247	23,763
June	18,070	45,332	21,350
March	19,974	50,030	23,706
May	17,783	44,456	21,081
November	21,358	53,364	25,270
October	16,902	42,342	19,958
September	17,463	43,826	20,388
1998			

### 5.2.2.1.11 Dice.

Formulario, que maneja las restricciones de las dimensiones del cubo (Figura 2.70).

Figura 2.70. Restricciones cubo.



## 6. PRUEBAS Y RESULTADOS

### 6.1 ANÁLISIS DE FUNCIONALIDAD

El análisis de funcionalidad de la herramienta se desarrolló utilizando un computador Intel(R) Core(TM) 2 CPU 4400 @ 2GHZ, Disco Duro de 250 GB, memoria RAM de 3 Gigas y tarjeta de video de 64MB. Para la realización de las pruebas de funcionalidad se hizo uso de la base de datos del sistema académico de la Universidad de Nariño, con el cual se construyó el datamart UDENAR.

### 6.2 DEFINICIÓN DE REQUISITOS PARA LA CONSTRUCCIÓN DEL DATAMART UDENAR

Después de analizar la estructura de la base de datos de la Universidad de Nariño, se identificó que cuenta con información del historial académico de los estudiantes. Consta de relaciones con información acerca de materias, notas, estudiantes, profesores y facultades.

Debido a la gran cantidad de registros de los estudiantes de la Universidad de Nariño, el análisis se limitará solamente a tomar datos de estudiantes del programa de Ingeniería de Sistemas en todas sus extensiones, que hayan ingresado a partir del periodo B del año 2003 hasta el periodo B del año 2008.

Del análisis del conjunto de datos objetivo se pudo definir cuatro métricas o indicadores que pueden ser de gran ayuda en el proceso de toma de decisiones en la facultad de ingeniería, en el programa de Ingeniería de Sistemas, además que estas métricas podrían ser tenidas en cuenta en el proceso de acreditación que se está llevando a cabo en el programa en este momento.

#### **Métricas:**

1. Cantidad de estudiantes del programa de Ingeniería de Sistemas.
2. Nota promedio final de las materias del pensum de Ingeniería de Sistemas.
3. Número de veces cursadas materias del pensum de Ingeniería de Sistemas.
4. Cantidad de eventos como (matriculas de honor, retiros, traslados y otros).

Definidas las métricas se realizara el diseño lógico del datamart, este estará compuesto por un conjunto de relaciones las cuales se catalogan en dos categorías:

- La primera categoría se denomina relaciones de hechos o más conocidas como las tablas “Fact”, estas tablas contendrán como atributos las métricas definidas anteriormente.
- La segunda categoría, son las relaciones de dimensión o las tablas “Dimensión”, estas tablas de dimensión son las que permiten cambiar la perspectiva del análisis de las métricas que se estén analizando.

A continuación se presenta una descripción de las relaciones de dimensiones del datamart.

- dim\_alumno: contiene datos acerca de los estudiantes.
- dim\_carrera: contiene datos referentes a las carreras.
- dim\_colegio: contiene datos a las concernientes a los colegios.
- dim\_especial: contiene datos acerca de casos especiales que se han presentado con los estudiantes.
- dim\_facultad: contiene datos a las concernientes a las facultades.
- dim\_materia: contiene datos referentes a las materias

En la tabla 2.7, se visualiza la descripción de las relaciones de dimensiones del datamart.

Tabla 2.7. Descripción de las tablas de dimensión del datamart UDENAR.

Relación	Atributo	Descripción Atributo
dim_alumno	cod_alumno	Código estudiante
	cod_colegio	Código colegio
	ciudad_nacimiento	Ciudad de nacimiento
	depto_nacimiento	Departamento nacimiento
	país_nacimiento	País nacimiento
	sexo	genero
	estado_civil	Estado civil
	barrio	Barrio
	estrato	Estrato
	No_hermano_univers	No tiene hermano en la Universidad
dim_carrera	cod_carrera	Código carrera
	cod_facultad	Código facultad
	nombre_carrera	Nombre de la carrera
	jornada	Tipo jornada
	periodo	periodo
	semestres	Numero            semestres



		carrera
	titulo_profesional	Título que se otorga
	ciudad	Ciudad donde se ofrece la carrera
dim_colegio	cod_colegio	Código colegio
	nombre_colegio	Nombre del colegio
	dep_colegio	Departamento donde se encuentra ubicado el colegio
	ciudad_colegio	Ciudad donde se encuentra ubicado el colegio
	jornada_colegio	Jornada del colegio
	calendario	calendario
	tipo_colegio	Tipo del colegio
dim_especial	cod_evento	Código del evento ocurrido
	nombre	Nombre del evento ocurrido
dim_facultad	cod_facultad	Código de la facultad
	nombre_facultad	Nombre de la facultad
	nombre_facultad_la	Nombre de la facultad completo
	decano	Nombre del decano
dim_materia	cod_materia	Código de la materia
	nombre	Nombre de la materia

A continuación, se presenta una descripción de las relaciones de hechos del datamart.

- fact\_aprobacion: contiene datos acerca de la nota final y veces cursadas.
- fact\_especial: contiene datos acerca de los eventos especiales.
- fact\_poblacion: contiene datos referentes a la población.

En la tabla 2.8 se presenta la descripción de las relaciones de hechos del datamart.

Tabla 2.8. Descripción de las tablas de hechos del datamart UDENAR.

fact_aprobacion	cod_alumno	Código del alumno
	cod_carrera	Código de la carrera
	cod_materia	Código de la materia
	nota	Nota final materia
	nveces	Número de veces vista

		materia
fact_especial	cod_alumno	Código del alumno
	cod_carrera	Código de la carrera
	cod_evento	Código de evento
	veces	Ocurrencia evento
fact_poblacion	cod_alumno	Código del alumno
	cod_carrera	Código de la carrera
	cant_alumno	Número de alumnos

En las figuras 2.71, 2.72 y 2.73 se presenta el modelado multidimensional del datamart UDENAR.

Figura 2.71. Esquema en copo de nieve, cantidad de eventos por región y carrera.

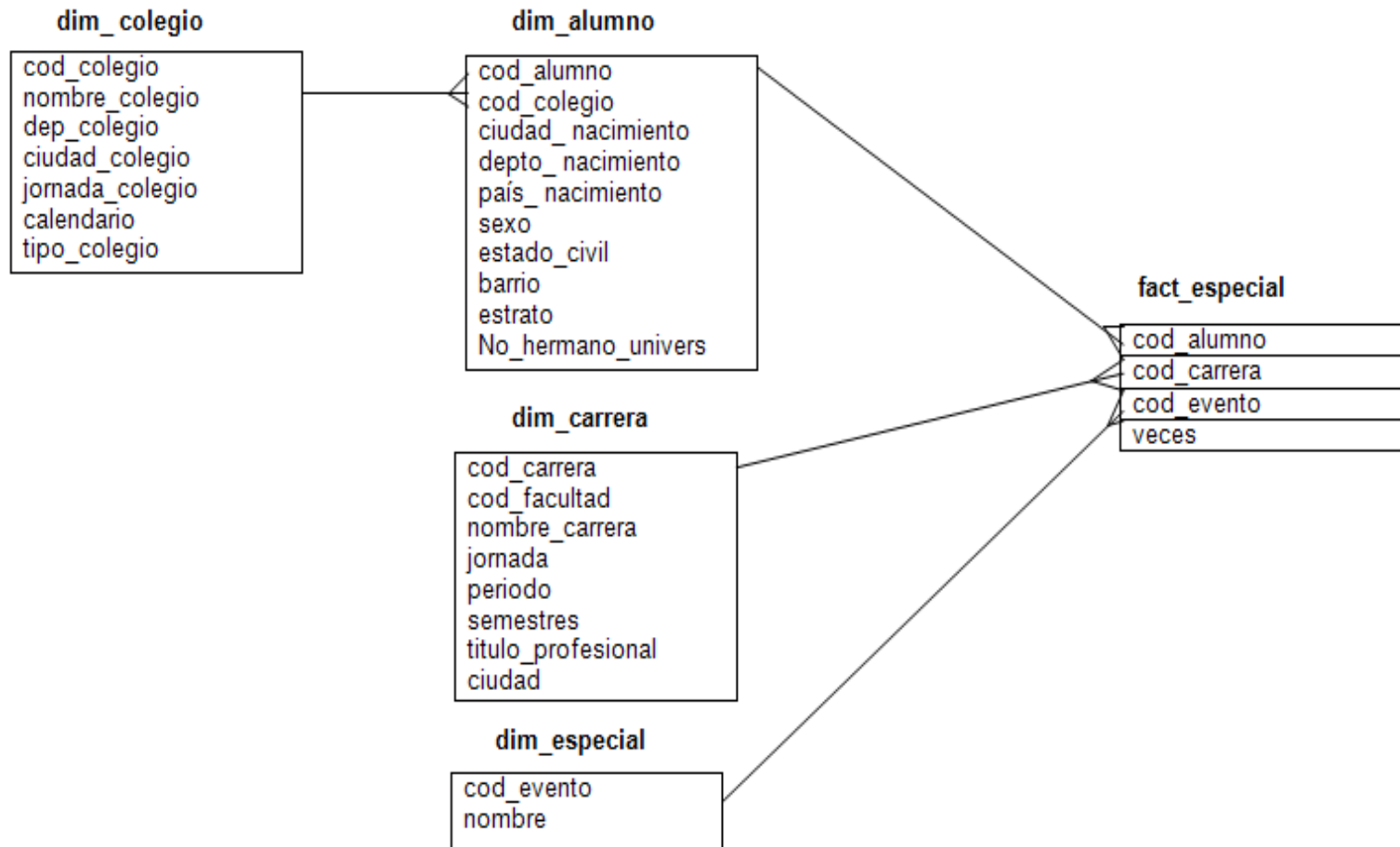


Figura 2.72. Esquema en copo de nieve, nota final y número de veces materia cursada por región y carrera.

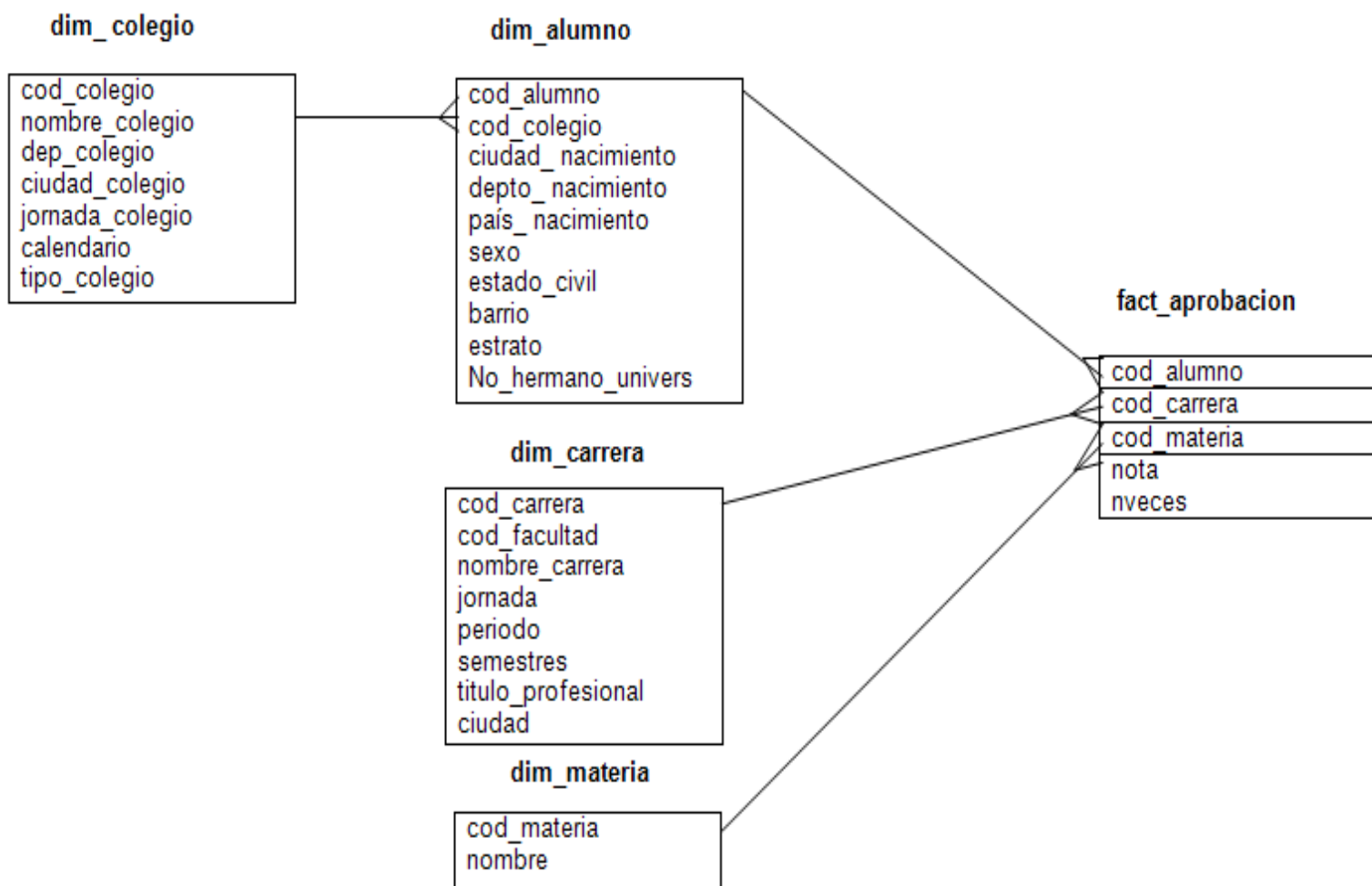
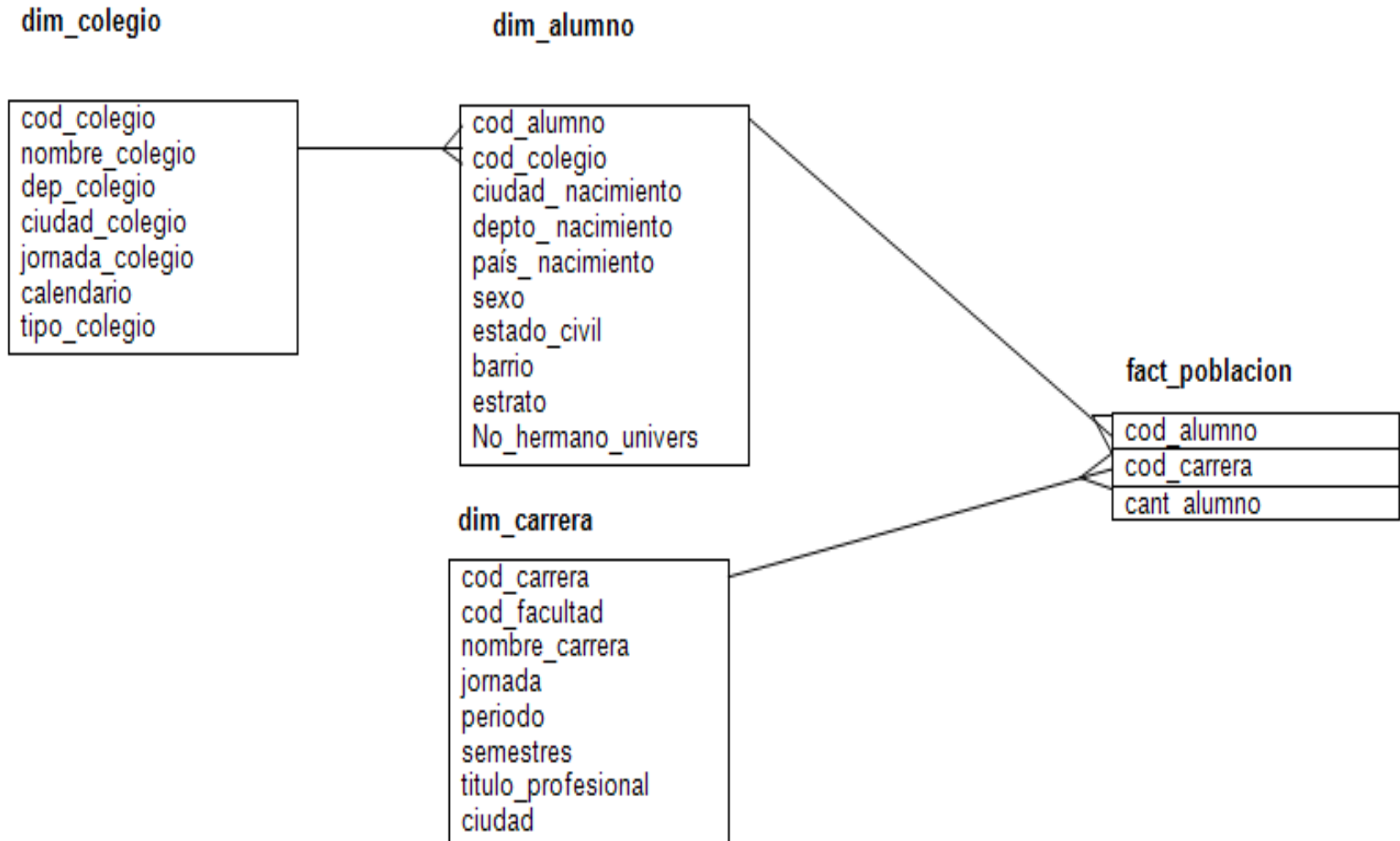


Figura 2.73. Esquema en copo de nieve, cantidad de alumnos por región y carrera



Una vez se tiene lista la estructura del datamart, se debe realizar una limpieza de los datos seleccionados para luego realizar la población del datamart.

### 6.3 CREACIÓN DE CUBOS

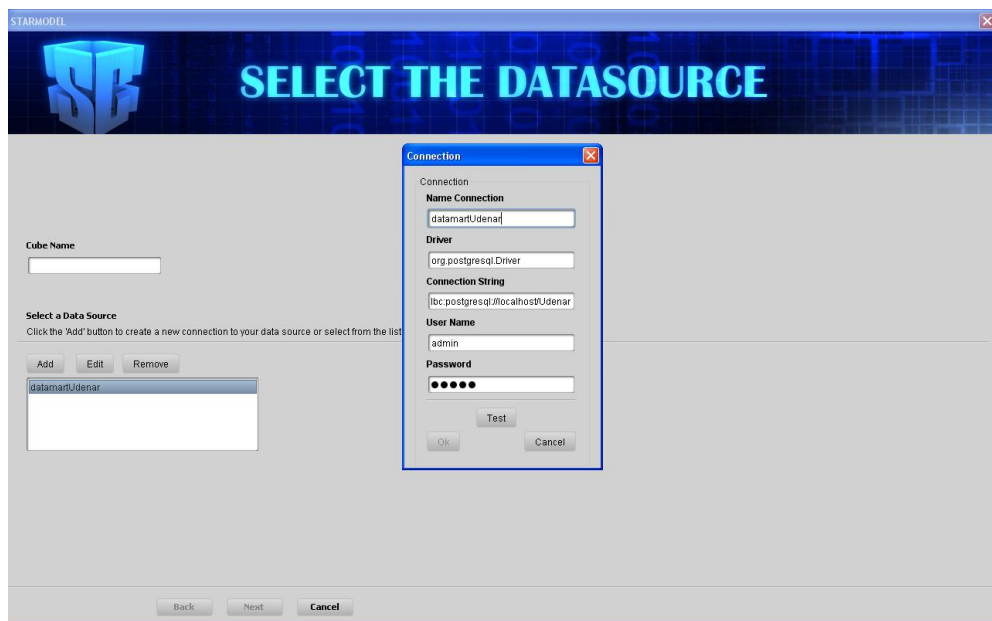
Para el análisis funcional se construyó 3 cubos a partir del datamart Udenar. La denominación de los cubos es la siguiente:

#### 6.3.1 Creación de cubo aprobación

- Cubo aprobación. Este cubo se construirá con el fin de determinar la nota final promedio y el máximo número de veces cursada una materia, de los estudiantes de Ingeniería de Sistemas de la Universidad de Nariño a partir del año 2003. A continuación se presenta la creación del cubo:

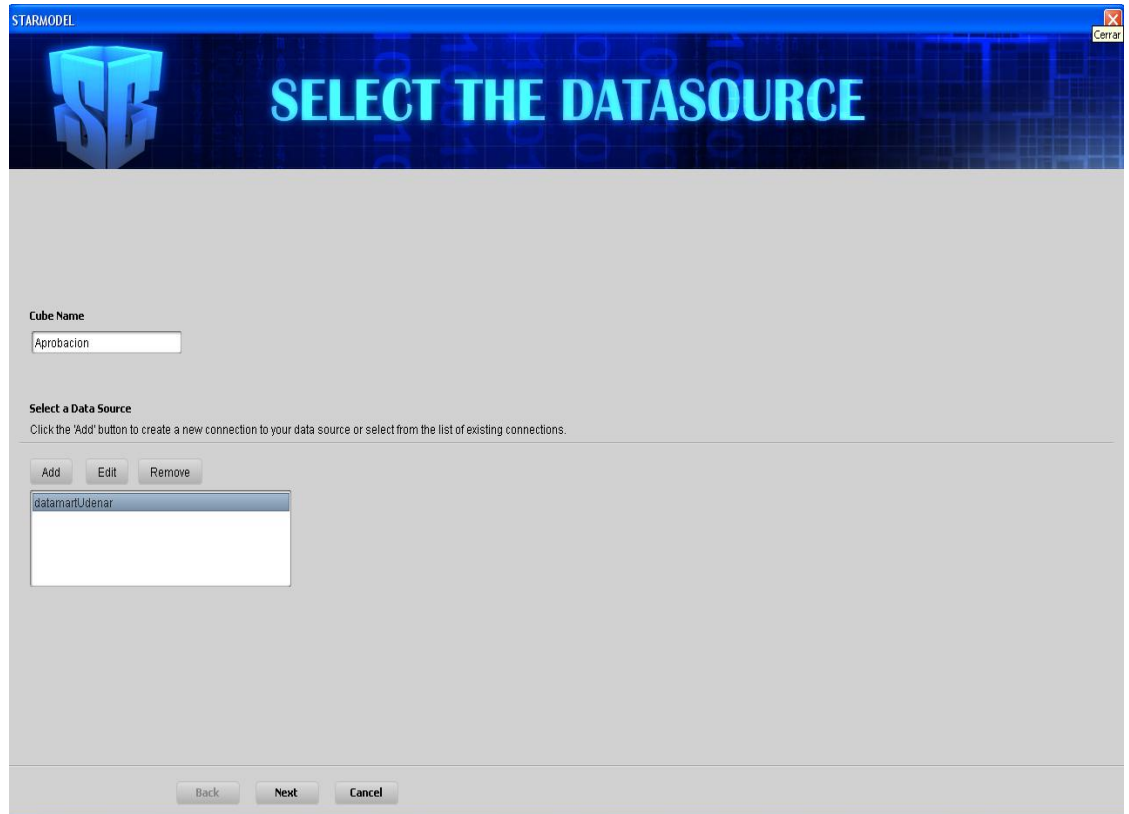
Primero: Se configura una fuente de datos, como se muestra en la figura 2.74.

Figura 2.74. Configuración conexión datamart.



Segundo: se le proporciona un nombre al cubo que se va a crear, como se presenta en la figura 2.75.

Figura 2.75. Nombre del cubo.



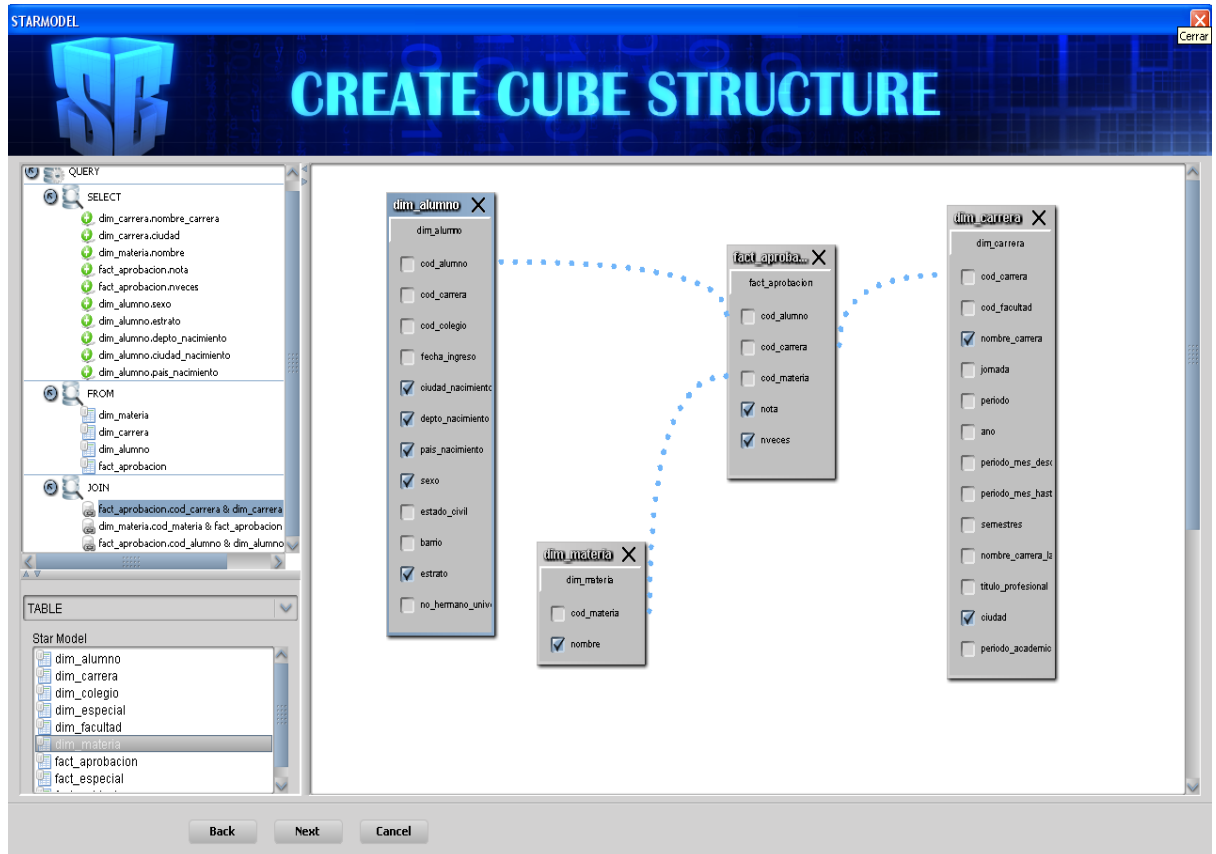
Tercero se seleccionan las tablas que se utilizarán en nuestro cubo, las cuales son:

- fact\_aprobacion. Esta corresponde a la tabla de hechos. Contiene las medidas: nota final promedio y número de veces que se han visto una materia.
- dim\_alumno. Esta tabla contiene la información completa de los alumnos.
- dim\_materia. Esta tabla contiene información acerca de las materias.
- dim\_carrera. Contiene información correspondiente a las carreras.

Se realizan los Join correspondientes y se seleccionan los siguientes campos:

De la tabla fact\_aprobacion: los campos nota y nveces; de la tabla dim\_materia: nombre; de la tabla dim\_alumno: ciudad\_nacimiento, depto\_nacimiento, pais\_nacimiento, sexo, estrato; y de la tabla dim\_carrera: nombre\_carrera y ciudad. El tercer paso se observa en la figura 2.76.

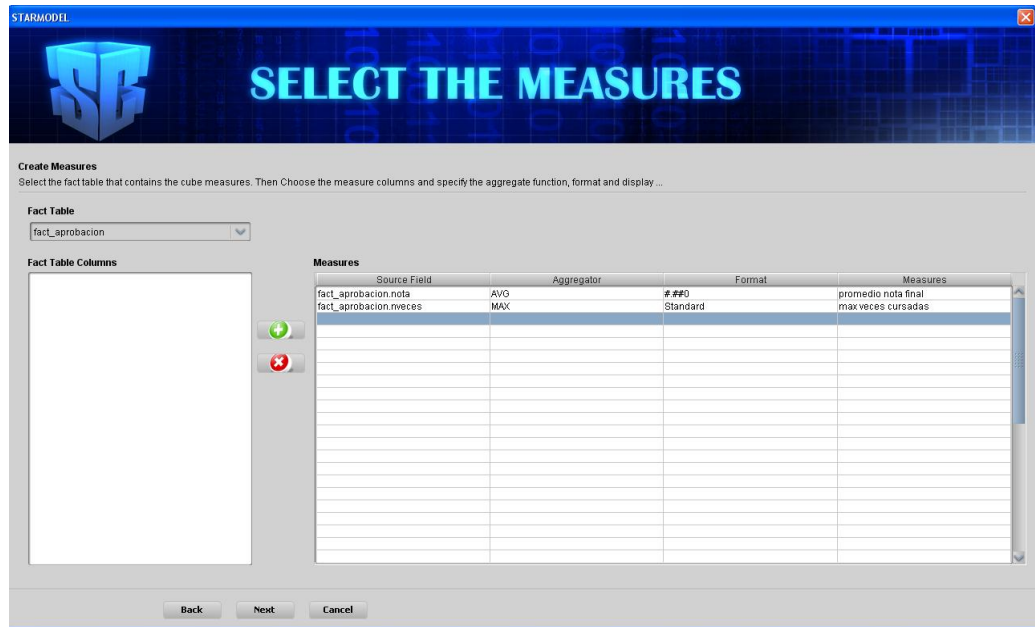
Figura 2.76. Conexiones y selección de campos.



Cuarto se selecciona la tabla de hechos: fact\_aprobacion y se agregan los hechos. Se cambian los operadores de agregación para cada fact. Para nota se escoge avg que corresponde al promedio de las notas finales. Para nveces se escoge max que significa el mayor número de veces cursada una materia. También se escoge el formato ###0 para nota que significa decimales y se le cambia el nombre a cada campo para visualizarlo, en la sección Measures. Este paso se presenta en la figura 2.77.

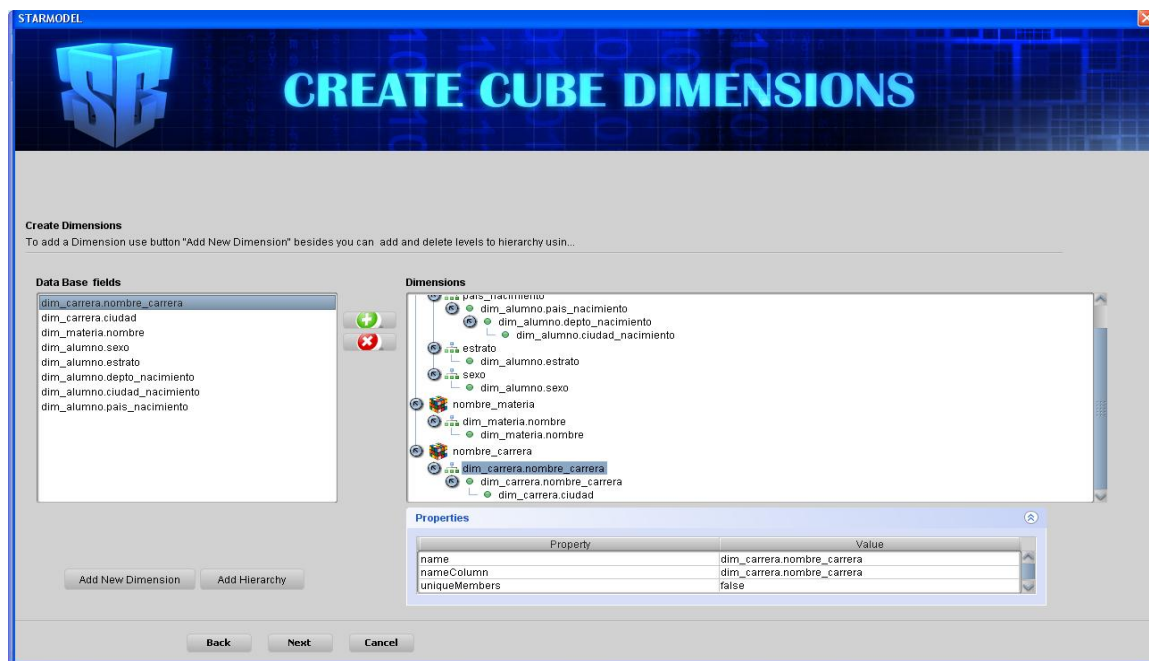


Figura 2.77. Definición de medidas.



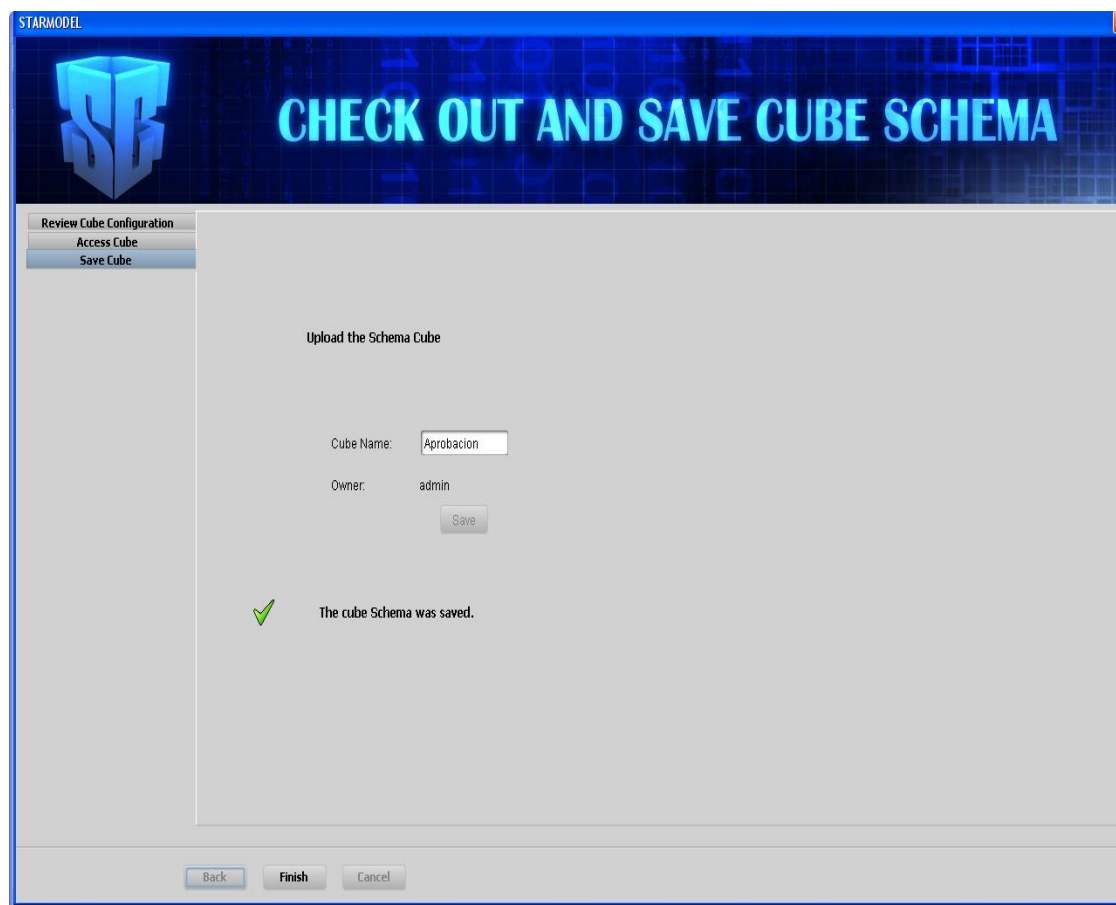
Quinto se crean las dimensiones a partir de las tablas dim\_carrera, dim\_alumno, dim\_materia. A partir de los campos seleccionados se crean las correspondientes jerarquías y sus niveles. Y se cambian los nombres de las jerarquías, como se visualiza en la figura 2.78.

Figura 2.78. Definición de dimensiones.



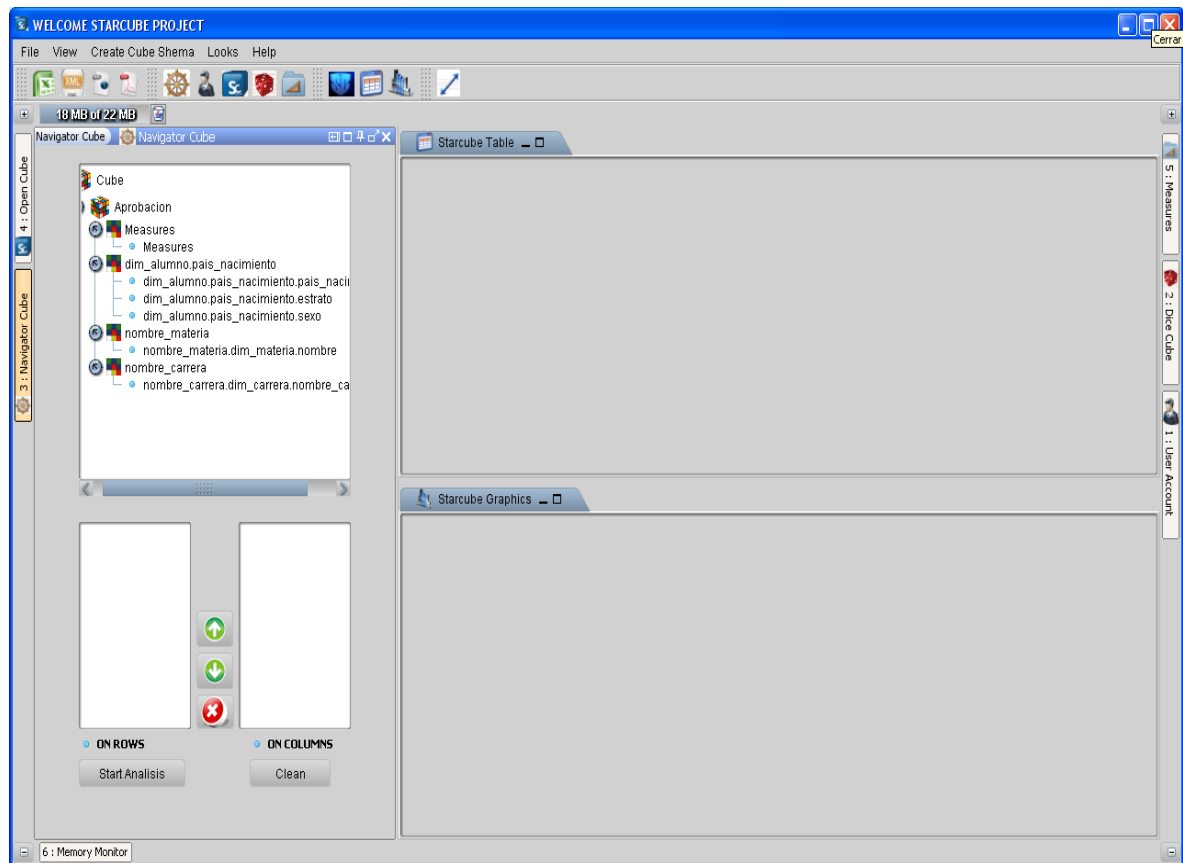
Para finalizar la creación de cubos se revisa que el archivo XML fue creado con éxito, se asignan privilegios de acceso y se guarda el cubo como se presenta en la figura 2.79.

Figura 2.79. Revisión del cubo.



La estructura final del cubo es la que se observa en la figura 2.80.

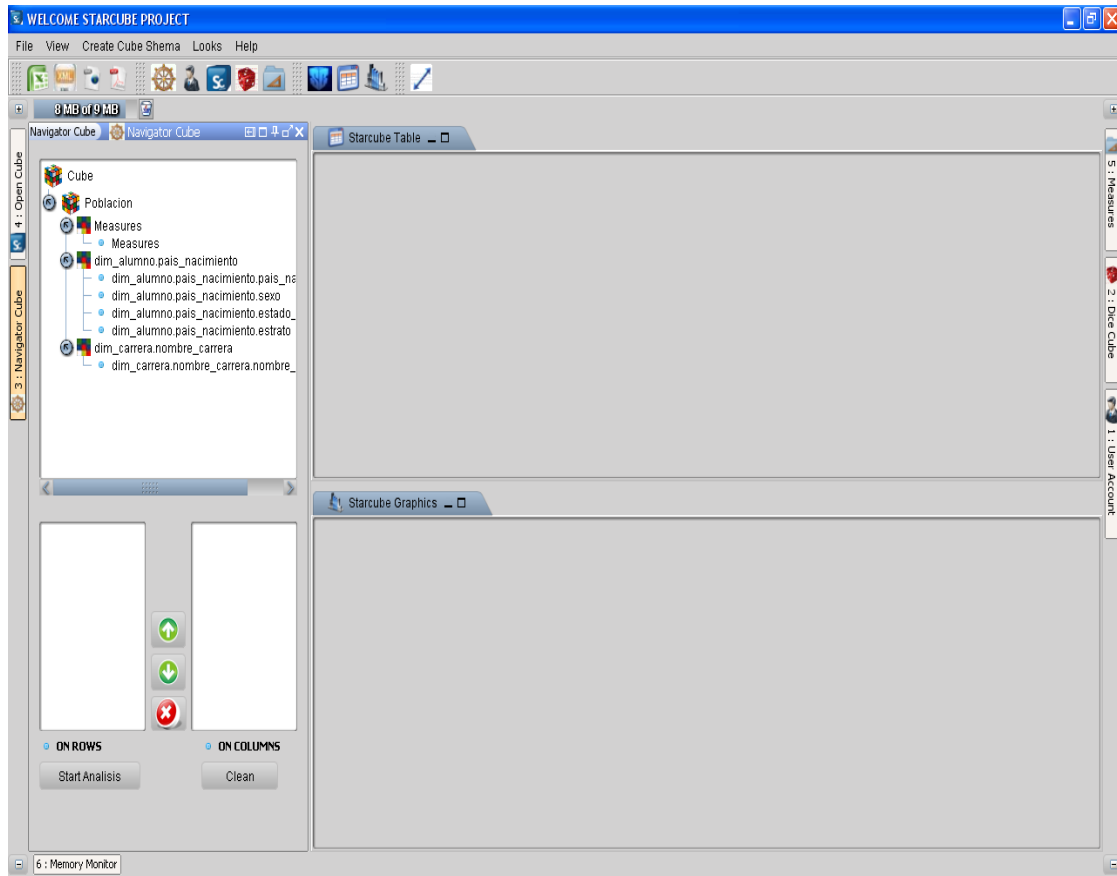
Figura 2.80. Estructura final cubo aprobación.



### 6.3.2 Creación de cubo población

- Cubo Población. Este cubo se construirá con el fin de determinar la población de los estudiantes de Ingeniería de Sistemas de la Universidad de Nariño y otras características a partir del año 2003. Como medida se tiene cantidad de alumnos. Como dimensiones se tienen: dim\_alumno y dim\_carrera. En la figura 2.81, se muestra la estructura del cubo.

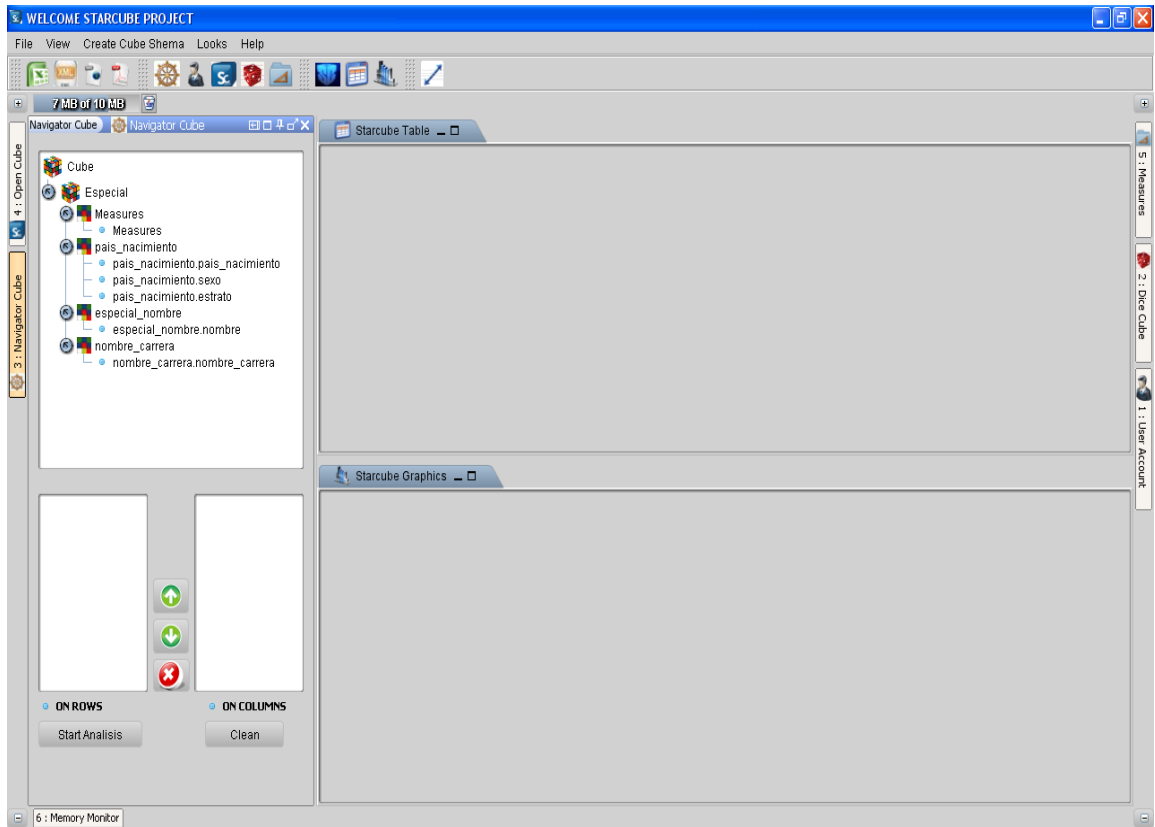
Figura 2.81. Estructura final cubo población.



### 6.3.3 Creación de cubo especial

- Cubo Especial. Este cubo se construirá con el fin de determinar los casos especiales que se han presentado con los estudiantes de ingeniería de Sistemas de la Universidad de Nariño a partir del año 2003, como retiros, traslados, pérdida de derecho a continuar estudios, matriculas de honor, entre otros. Como medida se tiene la cantidad de veces que se presentan estos casos especiales. Como dimensiones se tienen: dim\_pais, dim\_carrera y dim\_especial. En la figura 2.82 se muestra la estructura del cubo.

Figura 2.82. Estructura final cubo especial.

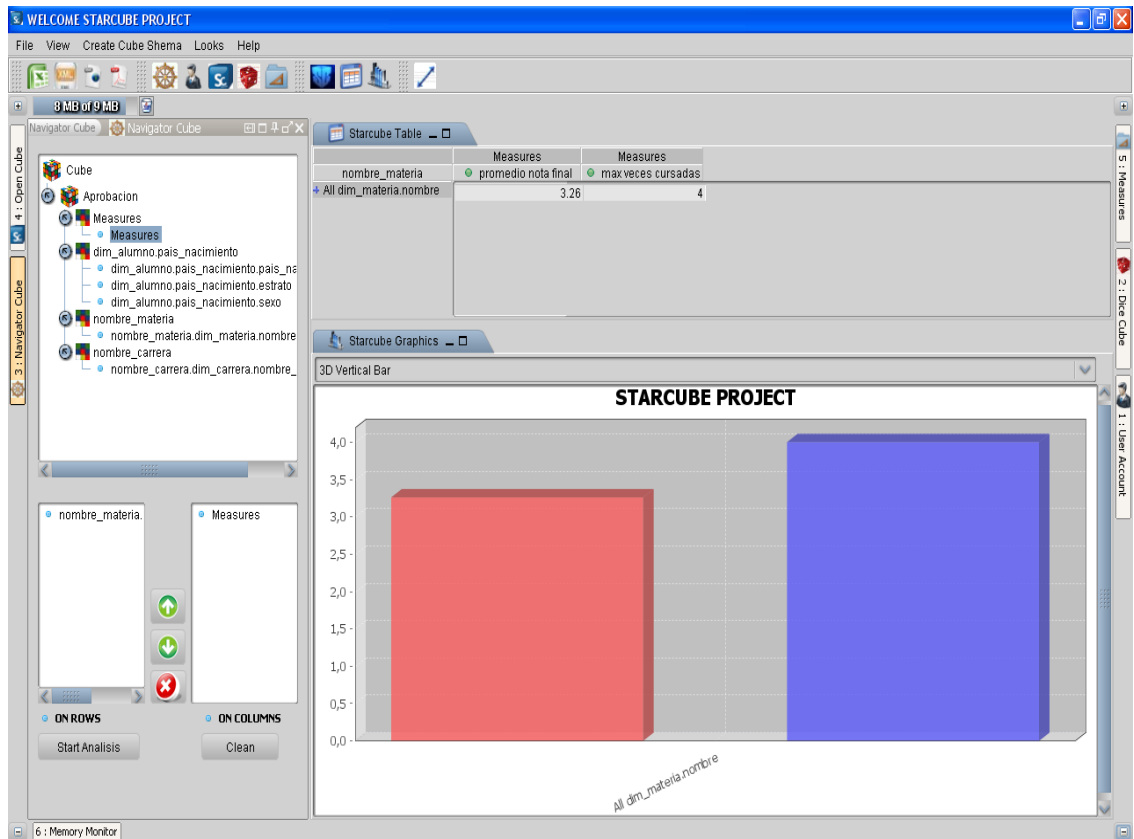


## 6.4 MANIPULACIÓN DE CUBOS

### 6.4.1 Manipulación de cubo aprobación

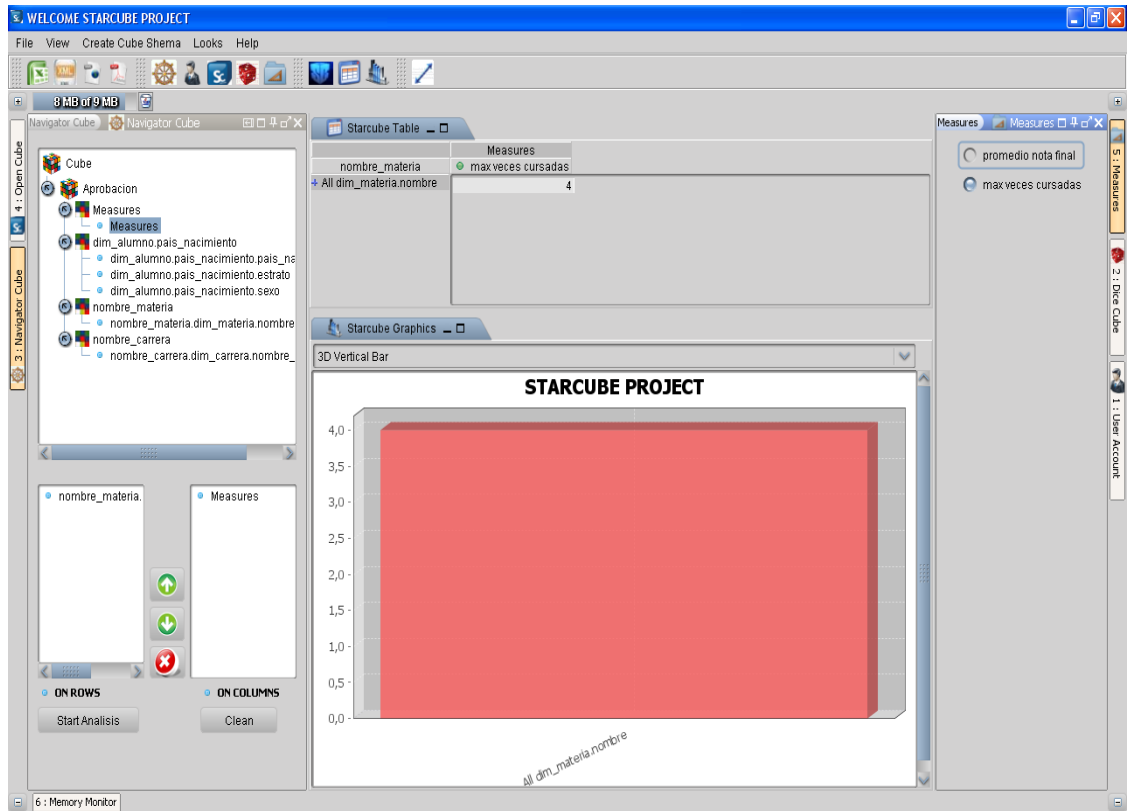
- Para la primera prueba se realiza la siguiente consulta:  
Se desea saber que materias son las que los estudiantes más veces las ven, es decir que materias son las que los estudiantes más las repiten.
  - Primero se abre el cubo aprobación ya que este cubo tiene como una de sus medidas el número de veces que ha sido vista una materia. Posteriormente, en el Navigator Cube, se arrastra al eje de rows la jerarquía nombre\_materia, perteneciente a la dimensión dim\_materia, y en el eje de cols se arrastra la jerarquía measures perteneciente a la dimensión de medida measures y se comienza el análisis con el botón Start Analisis. Este primer paso se observa en la figura 2.83.

Figura 2.83. Comenzar análisis.



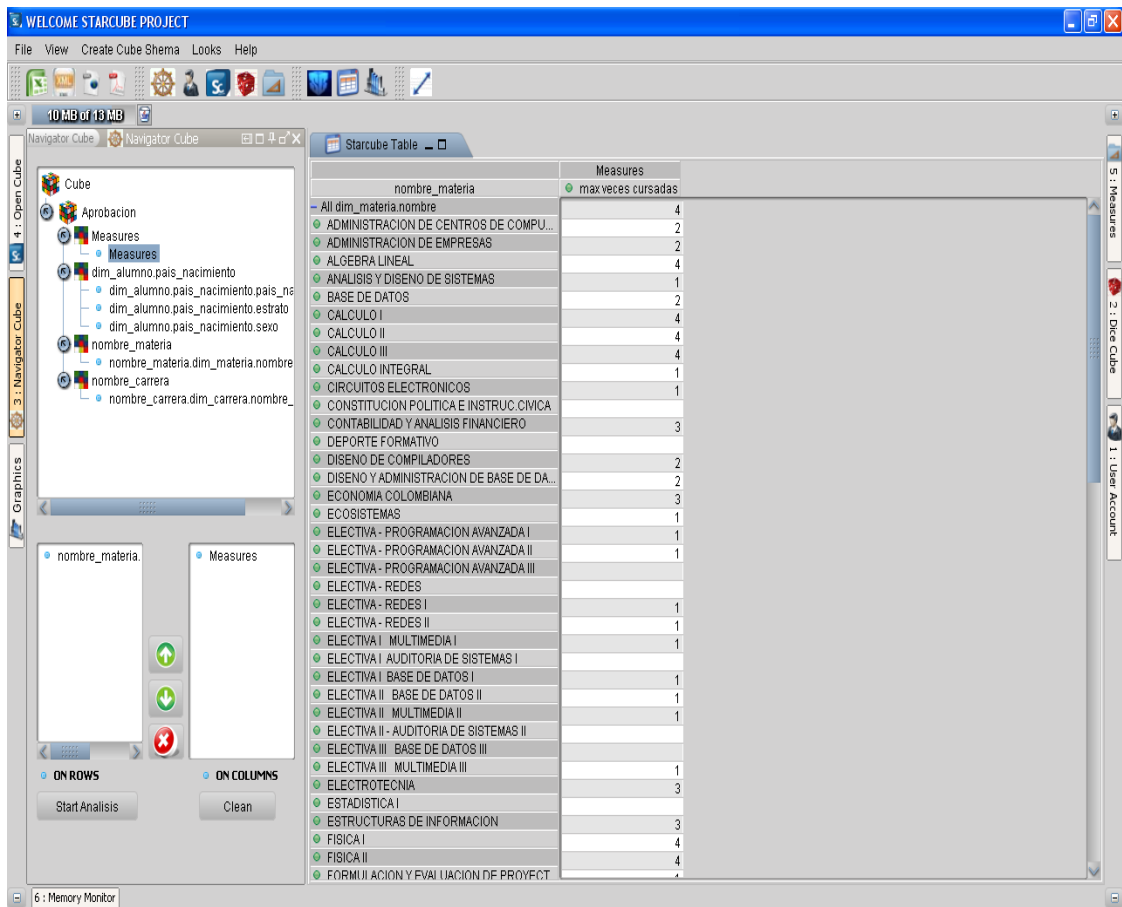
- Seguidamente se abre la pestaña de Measures para des chequear la medida promedio de la nota final, que no se la necesita como se visualiza en la figura 2.84.

Figura 2.84. Eliminar medida.



- En el análisis actual se observa, que en todas las materias se presenta el número de 4 veces cursadas. Pero la consulta inicial es saber que materias son las que presentan este caso, por lo que se procede a navegar en la estructura jerárquica del eje de rows en su dimensión materias. Para realizar esto se da clic normal en el elemento superior, el elemento tiene una cruz azul en la parte izquierda, lo que significa que tiene elementos inferiores a él, como se presenta en la figura 2.85.

Figura 2.85. Drill down.



Se presenta la lista de elementos inferiores al elemento que se dio clic, inmediatamente la cruz que este elemento tiene pasa a ser un menos, que significa los elementos que están por debajo de él pertenecen a su estructura jerárquica y que son inferiores a él. Cuando un elemento tiene un punto en su izquierda, significa que es hoja dentro de su estructura jerárquica y por tanto no tiene elementos inferiores. En esta lista están todas las materias del pensum académico del programa de Ingeniería de Sistemas desde el año 2003 y el número de veces que como máximo han sido vistas.

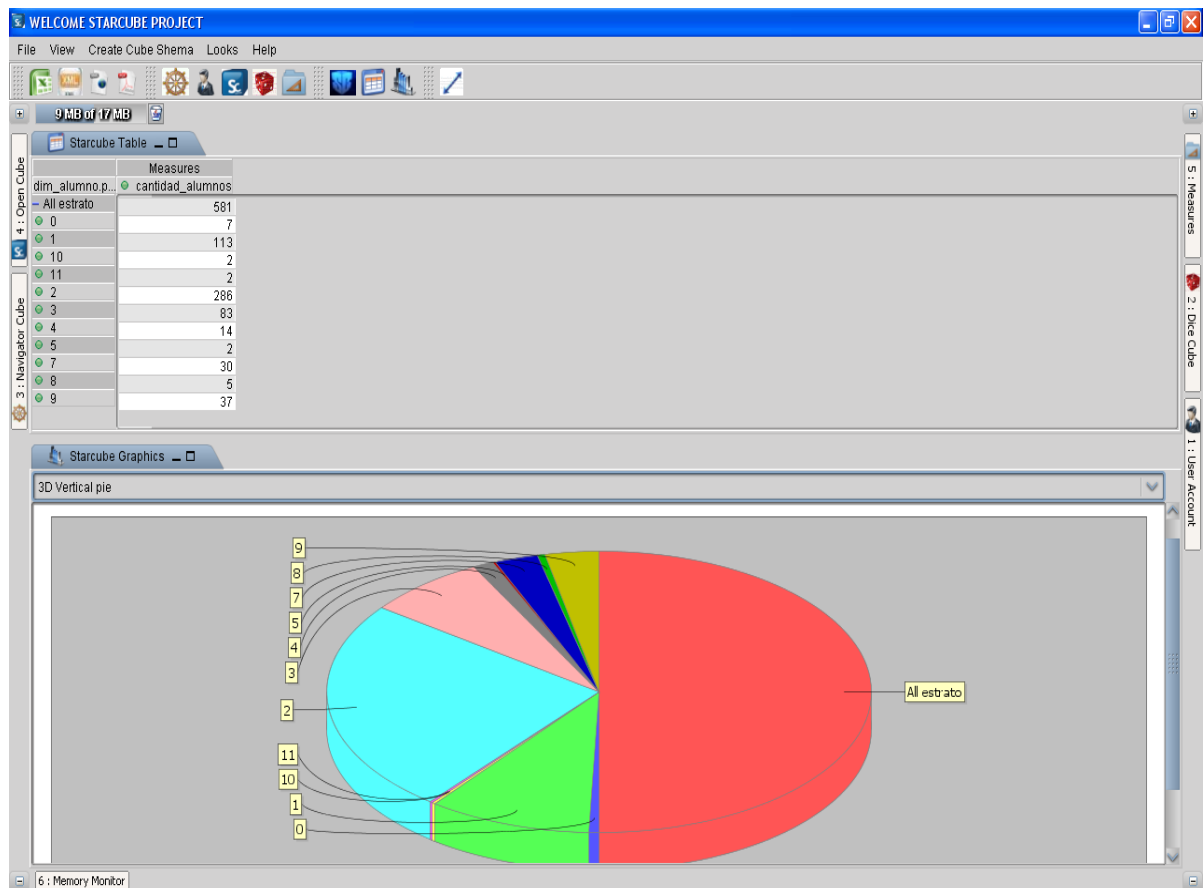
Se presentan entonces en éste análisis, que materias como álgebra lineal, Matemáticas generales, Cálculo I, Cálculo II, Cálculo III, Matemáticas especiales, Física I, Física II, entre otras, son materias que han presentado el caso de haber sido vistas 4 veces por algunos estudiantes. Con base en esta consulta se pueden realizar estudios de por qué se presenta este caso con estas materias, para así tomar decisiones posteriores que solventen este problema.



## 6.4.2 Manipulación de cubo población

- Para la segunda prueba se realiza la siguiente consulta:  
Se desea saber de los estudiantes de Ingeniería de Sistemas cuál es el estrato socioeconómico que predomina.  
Para esta consulta se utiliza el cubo Población que nos presenta la información relacionada con los estudiantes, su estrato socioeconómico, el país de nacimiento, la ciudad de nacimiento, entre otros. En la figura 2.86, se observa el resultado final del análisis de la manipulación del cubo población

Figura 2.86. Resultado final análisis.

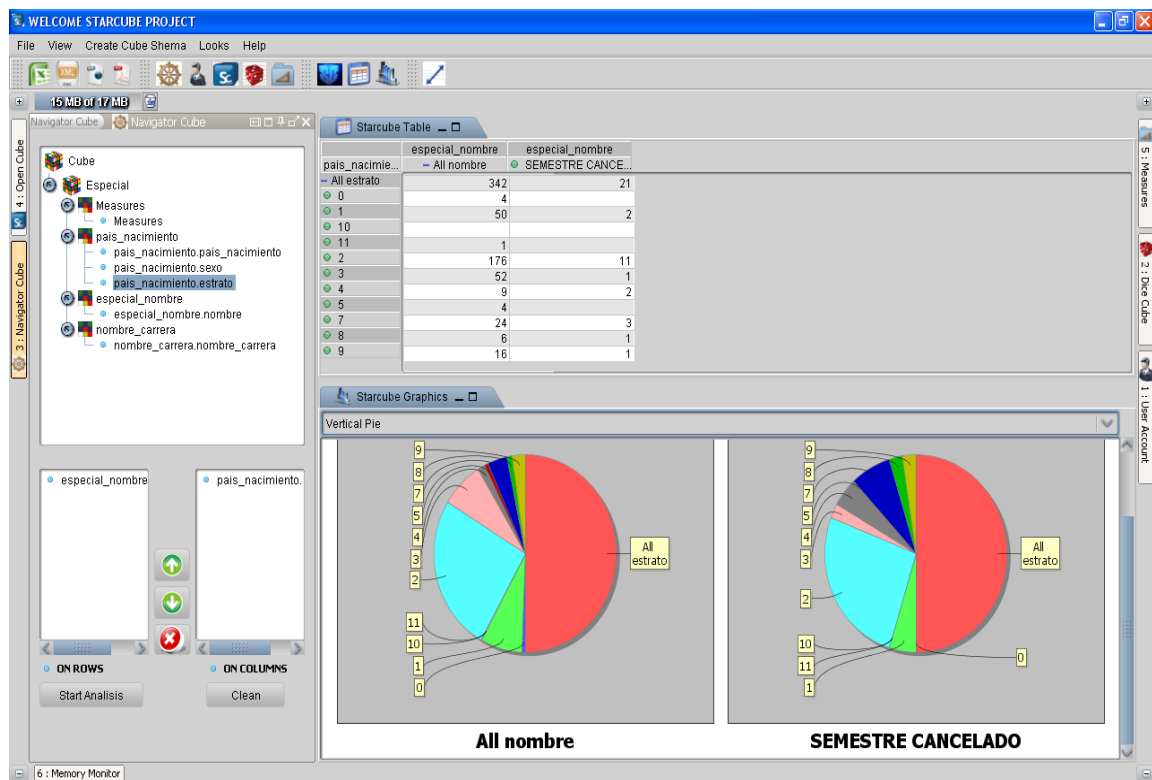


Se observa en el resultado de éste análisis que la mayor parte de estudiantes de Ingeniería de Sistemas proviene de los estratos 2 y 1 con 286 y 113 estudiantes respectivamente. Con base en estos resultados se tomarán las respectivas decisiones dependiendo del ámbito desde el cuál fue invocada esta consulta.

### 6.4.3 Manipulación del cubo especial

- Para la tercera prueba se realiza la siguiente consulta:  
Se desea saber si una de las posibles causas de la cancelación de semestres por parte de los estudiantes tiene que ver con el estrato socioeconómico de los implicados.  
Para esta consulta se utiliza el cubo Especial, que presenta información de casos especiales que se han presentado con los estudiantes del programa de Ingeniería de Sistemas, tales como cancelación de semestres, entre otros como se visualiza en la figura 2.87.

Figura 2.87. Resultado final análisis.



Este análisis entrega como resultado que la mayor parte de cancelaciones de semestre por parte de estudiantes se presenta en el estrato socioeconómico 2, dependiendo de este resultado se realizarán los correspondientes estudios para tratar de disminuir este problema.

## 7. CONCLUSIONES

1. Se presenta la primera versión de la herramienta “*STARCUBE: Una herramienta ROLAP de análisis multidimensional para el soporte a la toma de decisiones, débilmente acoplada con el SGBD PostgreSQL*”, como punto de inicio para futuras investigaciones.
2. Los resultados de las pruebas realizadas con la herramienta STARCUBE y utilizando el DataMart UDENAR, demuestran que ésta funciona correctamente. Las operaciones: Drill Down, Roll Up, Pivot, Slice y Dice, presentan resultados confiables y satisfactorios.
3. La arquitectura de STARCUBE conformada por los módulos: interfaz grafica de usuario (GUI), kernel y utilidades, hacen flexible esta herramienta, permitiendo su fácil actualización.
4. El desarrollo de STARCUBE con un lenguaje multiplataforma como lo es JAVA™ permite su portabilidad.
5. Por el hecho de usar utilitarios bajo licencia CPL (Common Public License) como lo son OLAP4J y Mondrian, hace que STARCUBE quede bajo este mismo tipo de licencia.
6. El proyecto permitió afianzar los conocimientos adquiridos durante la carrera de Ingeniería de Sistemas e investigar y profundizar en un tema tan interesante y actual como lo es el Procesamiento Analítico en Línea OLAP.

## **8. RECOMENDACIONES**

- 1.** Probar la herramienta STARCUBE en un proyecto que involucre una PYME, con el fin de evaluar su utilidad en la toma de decisiones de carácter real en el campo laboral.
- 2.** Implementar en STARCUBE un módulo que permita la construcción y el desarrollo del proceso de ETL de una bodega de datos, convirtiéndola en una Suite de Business Intelligence.
- 3.** Utilizar STARCUBE en las electivas del área de Bases de Datos de la Universidad de Nariño, como herramienta didáctica y de apoyo.

## REFERENCIAS

- [1] Fernández, C.: Imprecisión e Incertidumbre en el Modelo de Datos Multidimensional: Aplicación a la Minería de Datos, Editorial Universidad de Granada, Granada, España, (2005).
- [2] Figueroa, G.J.: Implementación de un Almacén de Datos para una Base de Datos DB2 usando Instrucciones SQL: una Solución ROLAP. Proyecto de grado, Universidad Católica de Maule, Talca, Chile (2007).
- [3] Inmon W. H. y Hackathorn R. D. (1994) Using the DataWarehouse. Wiley-QED Publication.
- [4] Inmon, W.: Building the Data Warehouse, Wiley Computer, New York 2 edición, (1996).
- [5] JASPER REPORTS, <http://jasperforge.org/public.php>
- [6] JFREECHART <http://www.jfree.org/jfreechart/>
- [7] Miranda, J.: Introducción a la Minería de Datos: Una perspectiva analítica, Departamento de Ingeniería Industrial, Universidad de Chile, Chile.
- [8] MYDOGGY, <http://mydoggy.sourceforge.net/>
- [9] Object Oriented Design with Applications Benjamming Cummings 1991.
- [10] Object Oriented Modeling and Design Prentice Hall 1991.
- [11] Object-Oriented Software Engineering: A Use Case Driven Approach Addison-Wesley 1992.
- [12] OLAP BROWSER, <http://www.iqub.com/>.
- [13] INSTANTOLAP, <http://www.instantOLAP.com/pricing>
- [14] OLAP Council <http://www.OLAPcouncil.org/index.html>.

- [15] OLAP4J, [http://OLAP4j.svn.sourceforge.net/viewvc/OLAP4j/trunk/doc/OLAP4j\\_fs.html](http://OLAP4j.svn.sourceforge.net/viewvc/OLAP4j/trunk/doc/OLAP4j_fs.html)
- [16] OLAPX, <http://www.OLAPxsoftware.com/default.asp>.
- [17] Pentaho Corporation <http://www.pentaho.com/index.php>.
- [18] SWINGX, <http://swinglabs.org/>
- [19] Timarán, R.: Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de datos: un Estado del Arte. En revista Ingeniería y Competitividad, Universidad del Valle, Volumen 3, No. 2, Cali, Colombia (2001).
- [20] Zvenger, P.: Introducción al Soporte de Decisiones: Incorporación de soluciones OLAP en entornos empresariales, Universidad Nacional del Sur, Argentina, (2005).

## ANEXO 1

### MARCAS REGISTRADAS UTILIZADAS

- Procesador Intel®Core™ Duo, Copyright ©Intel Corporation.
- Impresora LASER Hewlett Packard™, Copyright © 2008 Hewlett-Packard Development Company, L.P.
- Sistema Operativo LINUX Fedora™ Core 7, Copyright © 2008 Red Hat, Inc.
- JAVA™, Copyright 1994-2008 Sun Microsystems, Inc. All rights reserved.
- NetBeans™ 6.0, Copyright (C) 1997 - 2007 Sun Microsystems, Inc. All rights reserved.
- PostgreSQL™ v. 8.2.6, Copyright © 1996 – 2008 PostgreSQL Global Development Group.
- PENTAHO © 2009, Pentaho Corporation.
- Microsoft® Office Excel © 2009.
- Adobe Systems Incorporated, Copyright 1984-2004.

## ANEXO 2

### 1. INTRODUCCIÓN A STARCUBE

#### 1.1 ¿QUE ES STARCUBE?

STARCUBE es una herramienta ROLAP, desarrollada por estudiantes pertenecientes al Grupo de Investigación GRIAS (Grupo de Investigación Aplicado a Sistemas) dirigido por el Dr. Ricardo Timaran Pereira, Ph.D. del Programa de Ingeniería de Sistemas de la Universidad de Nariño de la ciudad de Pasto (Nariño – Colombia).

STARCUBE es un software de apoyo a toma de decisiones, mediante la creación de cubos de datos y su manipulación.

STARCUBE permite realizar todas las operaciones de OLAP como son dice, slice, dril down, roll up, pivot , todas esta operaciones para cambiar la perspectiva del análisis de los cubos de datos.

STARCUBE está continuamente en desarrollo y en constante optimización.

Para comentarios o sugerir ideas para mejorar, por favor escribimos a [starcube.project@gmail.com](mailto:starcube.project@gmail.com).

#### 1.2 REQUERIMIENTOS

Para el correcto funcionamiento de la herramienta se necesita tener instalado el siguiente software:

- PostgreSQL 8.3 o superior
- La Maquina Virtual de Java jre6 o superior
- Usuario del SGBD admin que tenga permiso de creación de base de datos, superusuario, puede crear roles.
- Crear y restaurar la base de datos StarCube con la utilización de archivo StarCube.backup.
- Instalador de STARCUBE v1.0



## 1.3 INSTALACIÓN

Para esto es necesario el archivo de instalación **Setup STARCUBE.exe** y seguir los siguientes pasos:

- Hacer doble clic en el archivo
- Seguir las instrucciones del instalador
- Aceptar la licencia CPL (Common Public License)
- Hacer clic en install para comenzar la instalación, esperar a que instale todos los archivos necesarios y Ejecutar la aplicación (para esto es necesario los requerimientos no incluidos).

## 2. ENTORNO DE LA HERRAMIENTA CON EJEMPLO PASO A PASO

### ¿Cómo iniciar sesión en STARCUBE?


Para iniciar sesión en STARCUBE es necesario que se tenga una cuenta en el SGBD PostgreSQL. Cuando ud. sea capaz de acceder en ella, introduzca su nombre de usuario y contraseña (los mismos que en el SGBD) para el inicio de sesión.

Para iniciar en otra url tan sólo cambie el archivo “server.properties” que se encuentra en el directorio de instalación de STARCUBE. Abra el archivo en un editor de texto y cambie el valor de la variable “server” (Ver figura 1).

Figura 1. Inicio de Sesión en STARCUBE



## MANEJO DE USUARIOS

El manejo de usuarios es un asunto importante que STARCUBE provee para mantener sus datos libres de intrusos y más seguros. Para ir al panel “User Account” presione el botón  en la barra de herramientas, o haga clic en la

pestaña de la parte derecha de la ventana.

En el panel “User Account” se muestran los diferentes usuarios que están registrados en la base de datos de STARCUBE. Aquí se puede añadir, editar o eliminar usuarios. También existe un botón de refresco que actualiza la lista (Ver figura 2).

Figura 2. Manejo de usuarios

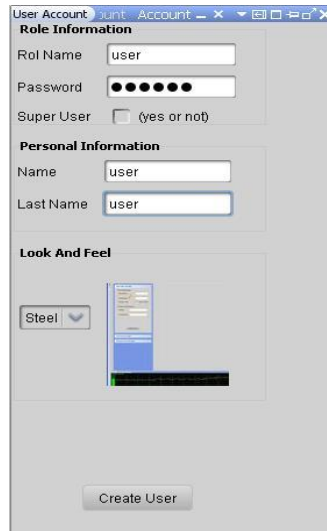


### **Añadir, editar y eliminar un usuario**

#### **Añadir un usuario**

Cuando se quiera añadir un nuevo usuario a la base de datos STARCUBE, tan sólo escriba su nombre de usuario y contraseña, escoja si éste tendrá permisos de Súper Usuario haciendo clic en la casilla de verificación, llene la información personal y seleccione el Look&Feel preferido (Apariencia de STARCUBE) (Ver figura 3).

Figura 3. Operación añadir usuario




The screenshot shows a window titled 'User Account' with a subtitle 'Account'. It is divided into three sections: 'Role Information', 'Personal Information', and 'Look And Feel'. In the 'Role Information' section, 'Rol Name' is 'user', 'Password' is masked with dots, and 'Super User' is unchecked. In the 'Personal Information' section, 'Name' and 'Last Name' are both 'user'. In the 'Look And Feel' section, a dropdown menu shows 'Steel' and a preview window displays a blue-themed interface. A 'Create User' button is at the bottom.

### Editar un usuario

Para editar un usuario existente reescriba sus parámetros de la misma manera en que se añade un usuario (rol, contraseña, datos personales y su look&feel) (Ver figura 4).

Figura 4. Operación editar usuario

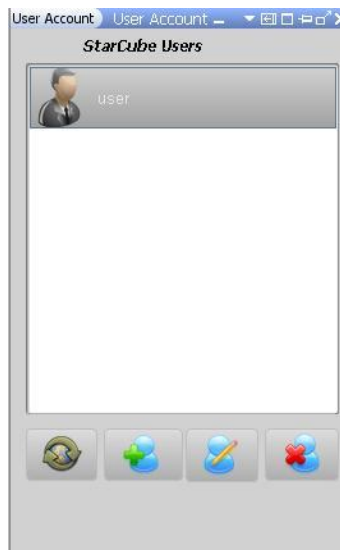


The screenshot shows the same 'User Account' dialog box, but for editing. The 'Role Information' section has 'Rol Name' as 'user', 'Password' as an empty field, and 'Super User' checked. The 'Personal Information' section has 'Name' as 'user' and 'Last Name' as 'user'. The 'Look And Feel' section has a dropdown menu showing 'Spr...' and a preview window displaying a blue-themed interface. A 'Create User' button is at the bottom.

## Eliminar un usuario

Para eliminar un usuario vaya al panel “User Account”, seleccione el usuario que desea remover y presione el botón “eliminar” en la parte inferior derecha del panel (Ver figura 5).

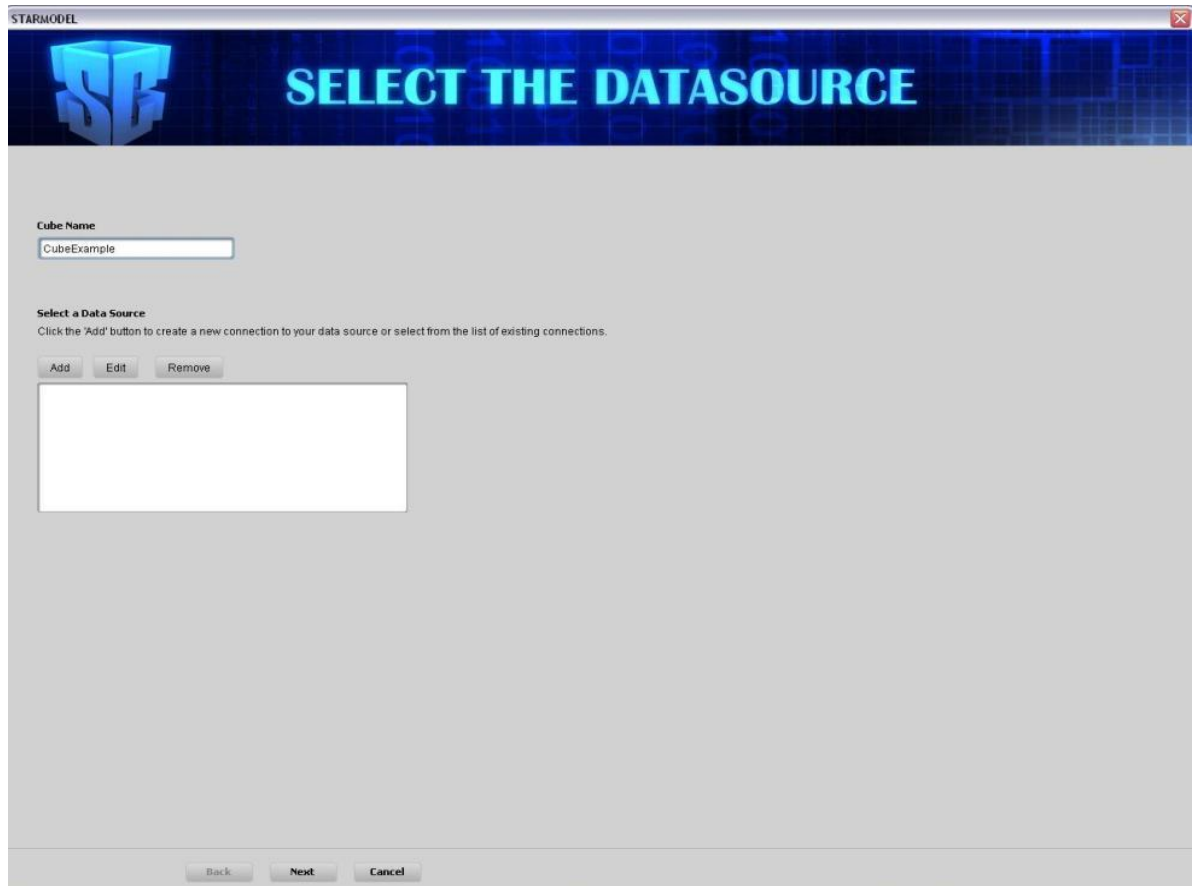
Figura 5. Operación eliminar usuario



## CREAR EL ESQUEMA DEL CUBO

Para realizar este paso vaya al Menú “Create Cube Schema”, luego escoja la opción “Configure Cube”. En esta ventana será posible empezar a construir un Cubo de datos. Para comenzar, tan sólo escriba el nombre que quiera para el Cubo en el campo de texto denominado “Cube Name”. El esquema del cubo será guardado en un archivo con formato XML (Ver figura 6).

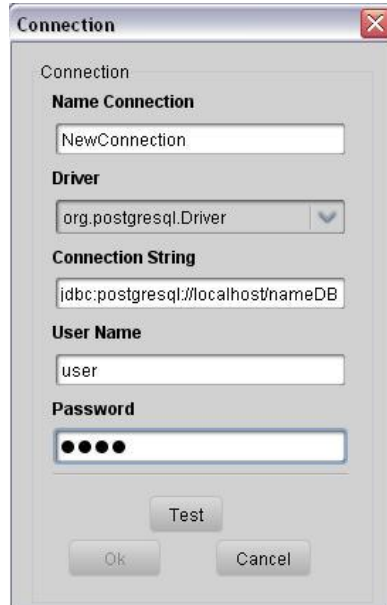
Figura 6. Selección de la fuente de datos



### **Añadir una conexión a la base de datos**

Para añadir una conexión a la base de datos, presione el botón "Add", luego en la ventana que se presenta escriba el nombre de la conexión, escoja el driver, coloque el nombre de la fuente de base de datos y por último escriba el usuario y contraseña del usuario PostgreSQL (Ver figura 7).

Figura 7. Operación añadir / editar una conexión



Para que la conexión quede bien configurada presione el botón “Test”. Aparecerá una alerta diciendo si la conexión fue exitosa. Para finalizar presione el botón “ok” (Ver figura 8).

Figura 8. Conexión exitosa



### Editar una conexión

Para editar cualquier conexión localizada en la lista, tan sólo selecciónela y después haga clic en el botón “Edit” para personalizarla. Aparecerá la misma ventana que cuando se añade una conexión, donde podrá cambiar su configuración (Ver figura 7).

## Eliminar una conexión

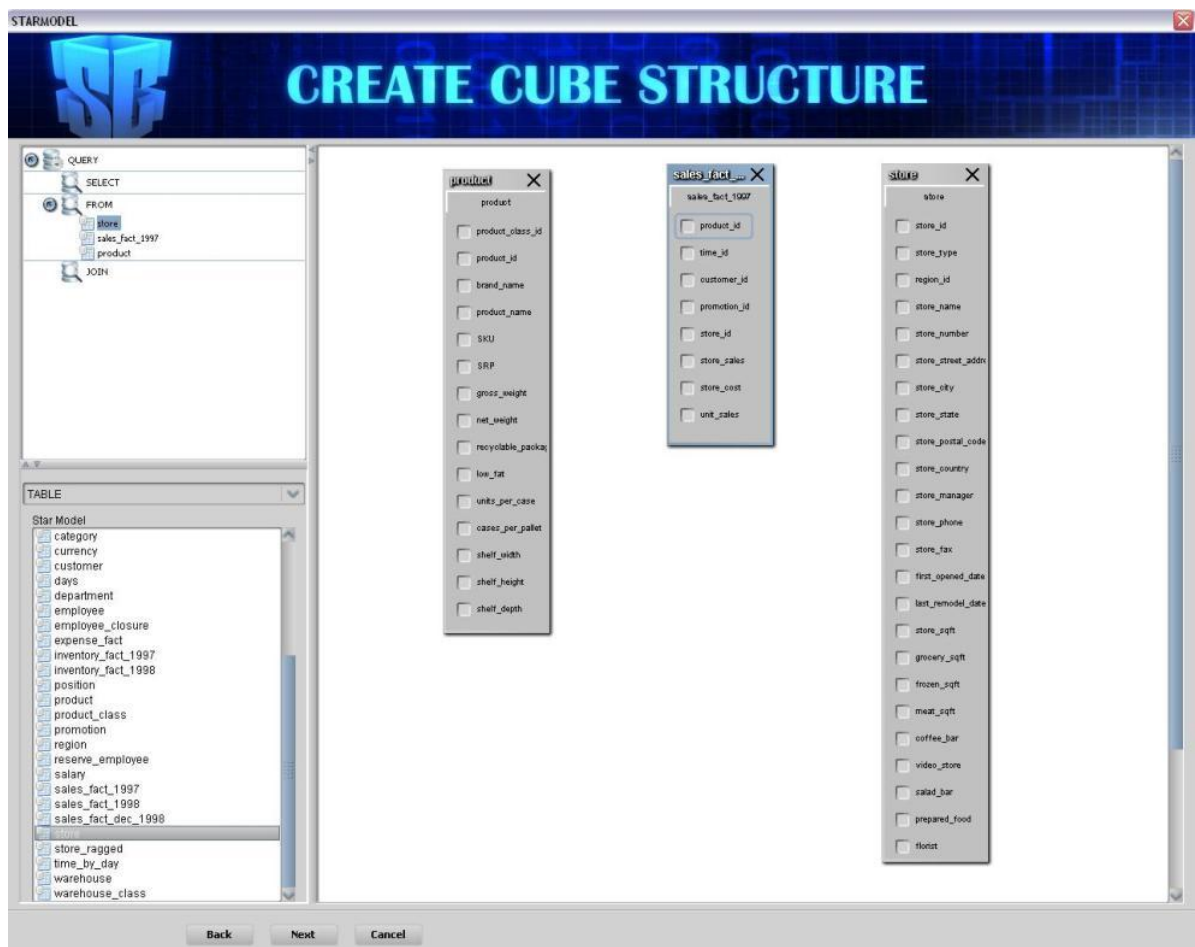
Para remover una conexión existente en la lista, selecciónela y luego haga clic en el botón “Remove” (ver Figura 6).

## CREAR LA ESTRUCTURA DEL CUBO

### Arrastrando tablas de hechos y dimensiones

Para crear el esquema del cubo se necesita arrastrar una Tabla de Hechos (medidas, valores con los que se puede operar) y Tablas de Dimensiones y luego soltarlos en el panel central de trabajo, preferiblemente colocando la Tabla de Hechos en el centro y las Tablas de Dimensiones a su alrededor para mayor comprensión (Ver figura 9).

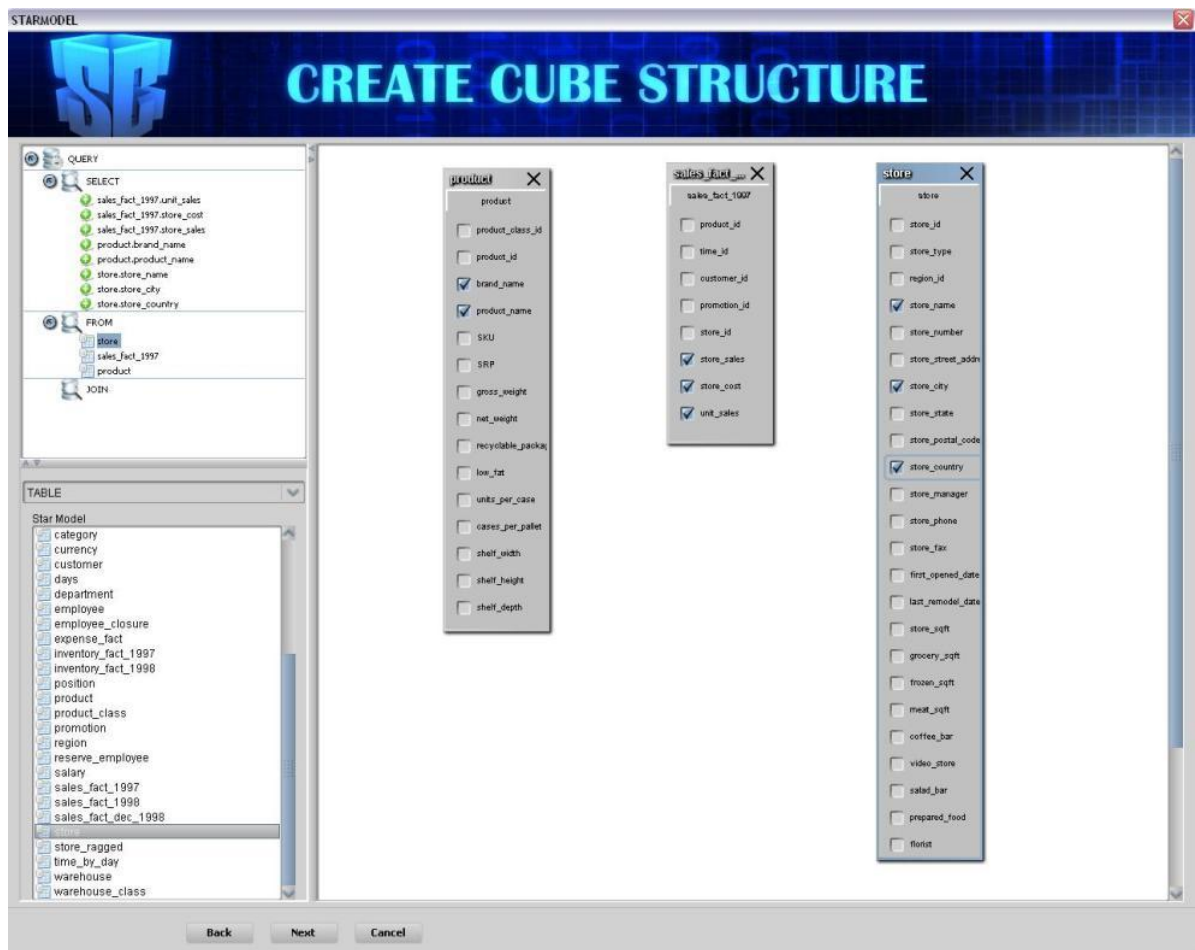
Figura 9. Creación de la estructura del cubo



## Selección de campos de las tabla de hechos y tablas de dimensiones

Para seleccionar los campos a analizar haga clic en las respectivas casillas de verificación. Notará que estas casillas cambian su estado, indicando que el campo fue escogido. Además se puede observar un árbol en el primer panel (lado izquierdo de la ventana) representando la consulta SQL con los campos seleccionados (Ver figura 10).

Figura 10. Selección de campos de las Tabla de Hechos y Tablas de Dimensiones



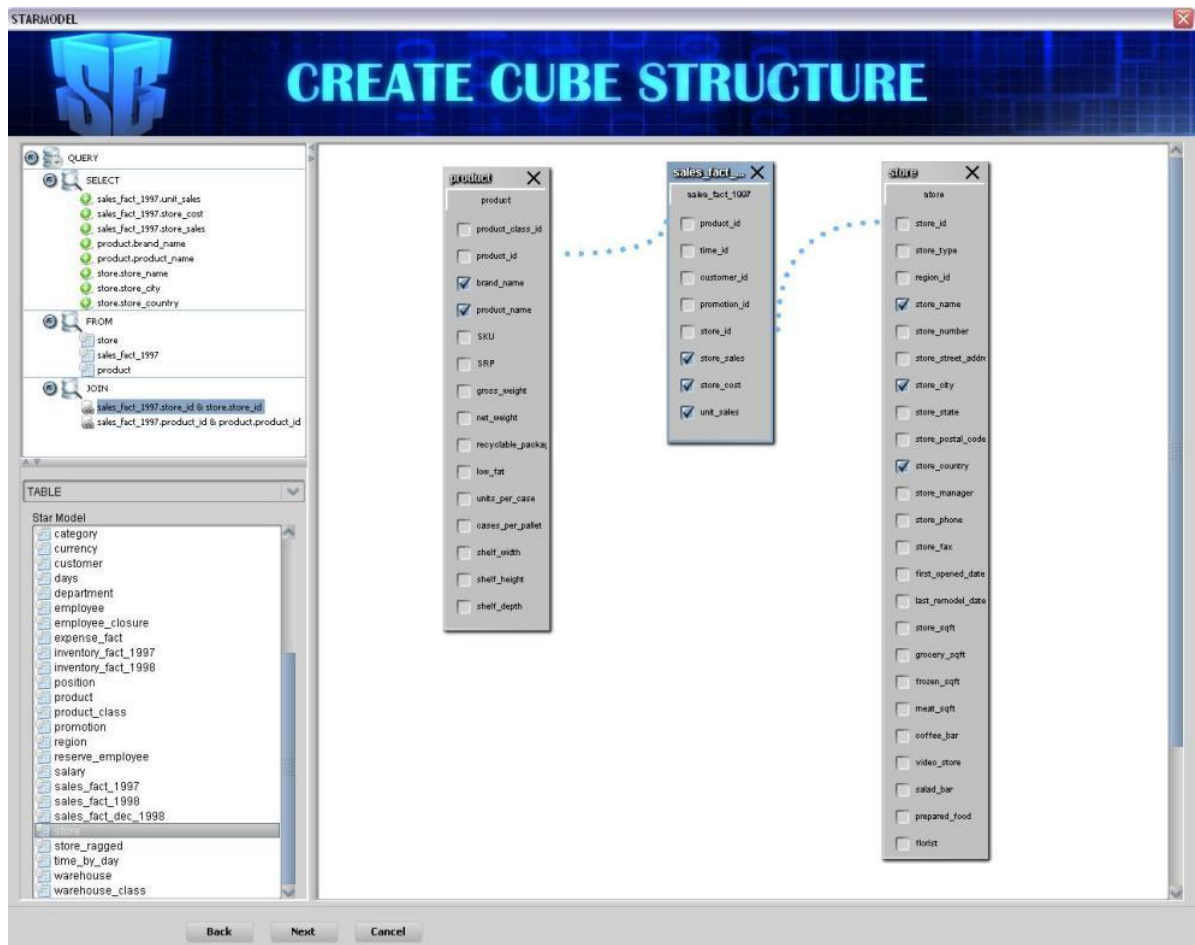


## Hacer un enlace entre tablas

Para hacer un enlace entre dos tablas, seleccione un campo, arrástrelo y suéltelo sobre otro campo de otra tabla, aparecerá una línea punteada mostrando el enlace. En este paso, se debe estar seguro que el enlace creado está hecho apropiadamente.


Este procedimiento también es representado en el panel izquierdo, donde se podrá eliminar cualquier enlace creado, haciendo clic derecho sobre el elemento que se quiere eliminar y seleccionando la opción "Delete Join" (Ver figura 11).

Figura 11. Creación de un enlace (JOIN) entre dos tablas.



## Seleccionar medidas o métricas

Para seleccionar las Medidas escogidas en el paso anterior, se debe dar clic en la

lista desplegable y seleccionar la Tabla de Hechos o “Fact Table”, luego, cuando sus campos aparezcan debajo, se escogen los valores que se quiere mostrar más tarde, haciendo clic en el botón . Las Métricas automáticamente aparecerán en la tabla del lado derecho.


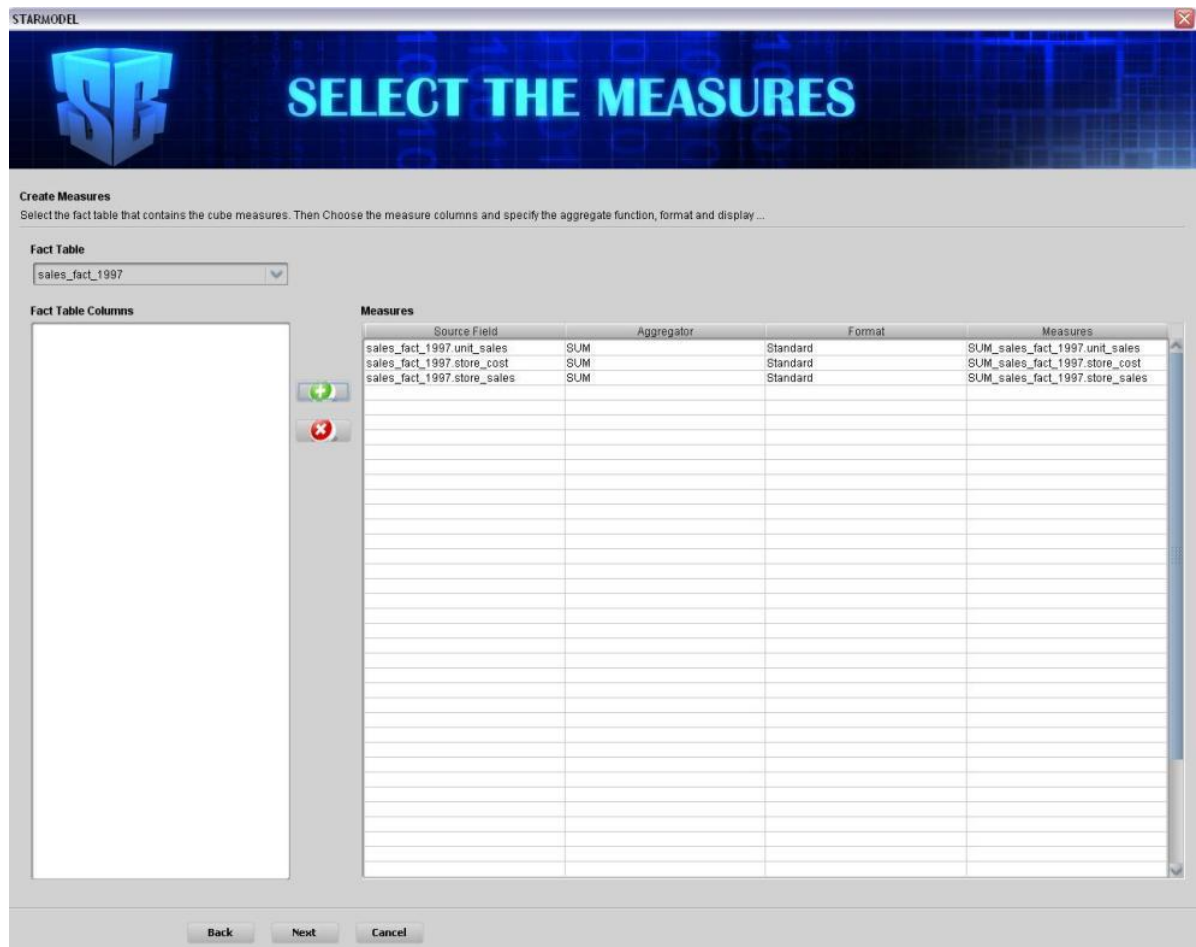
Para eliminar cualquier Medida de la tabla haga clic en el botón  (Ver figura 12).

Figura 12. Selección de medidas.



Una vez que se han escogido las Medidas a mostrar, se puede especificar:

- Funciones agregadas (Sum, Average, Count, Minimum, Maximum) (Ver figura 13).

Figura 13. Funciones agregadas

Source Field	Aggregator	Format	Measures
sales_fact_1997.unit_sales	SUM	Standard	SUM_sales_fact_1997.unit_sales
sales_fact_1997.store_cost	SUM	Standard	SUM_sales_fact_1997.store_cost
sales_fact_1997.store_sales	AVG	Standard	SUM_sales_fact_1997.store_sales
	COUNT		
	MIN		
	MAX		

- Formato numérico (Ver figura 14).

Figura 14. Formato numérico

Source Field	Aggregator	Format	Measures
sales_fact_1997.unit_sales	SUM	Standard	SUM_sales_fact_1997.unit_sales
sales_fact_1997.store_cost	SUM	Standard	SUM_sales_fact_1997.store_cost
sales_fact_1997.store_sales	SUM	#,##0	SUM_sales_fact_1997.store_sales


- Y la opción de renombrar los campos de las Medidas (Ver figura 15).

Figura 15. Nombre de las medidas

Source Field	Aggregator	Format	Measures
sales_fact_dec_1998.unit_sales	COUNT	Standard	UNIT_SALES
sales_fact_dec_1998.store_cost	SUM	Standard	STORE_COST
sales_fact_dec_1998.store_sales	SUM	Standard	STORE_SALES

## Crear dimensiones

En este paso se pueden crear Dimensiones con sus Jerarquías y Niveles.

Para crear una Dimensión haga clic en el botón “Add New Dimension”. Se creará la Dimensión y automáticamente una Jerarquía y un Nivel con el mismo nombre, pero se tendrá la opción de añadir nuevas Jerarquías a la Dimensión haciendo clic en el botón “Add Hierarchy”, o un nuevo Nivel a las Jerarquías, haciendo clic en el botón .


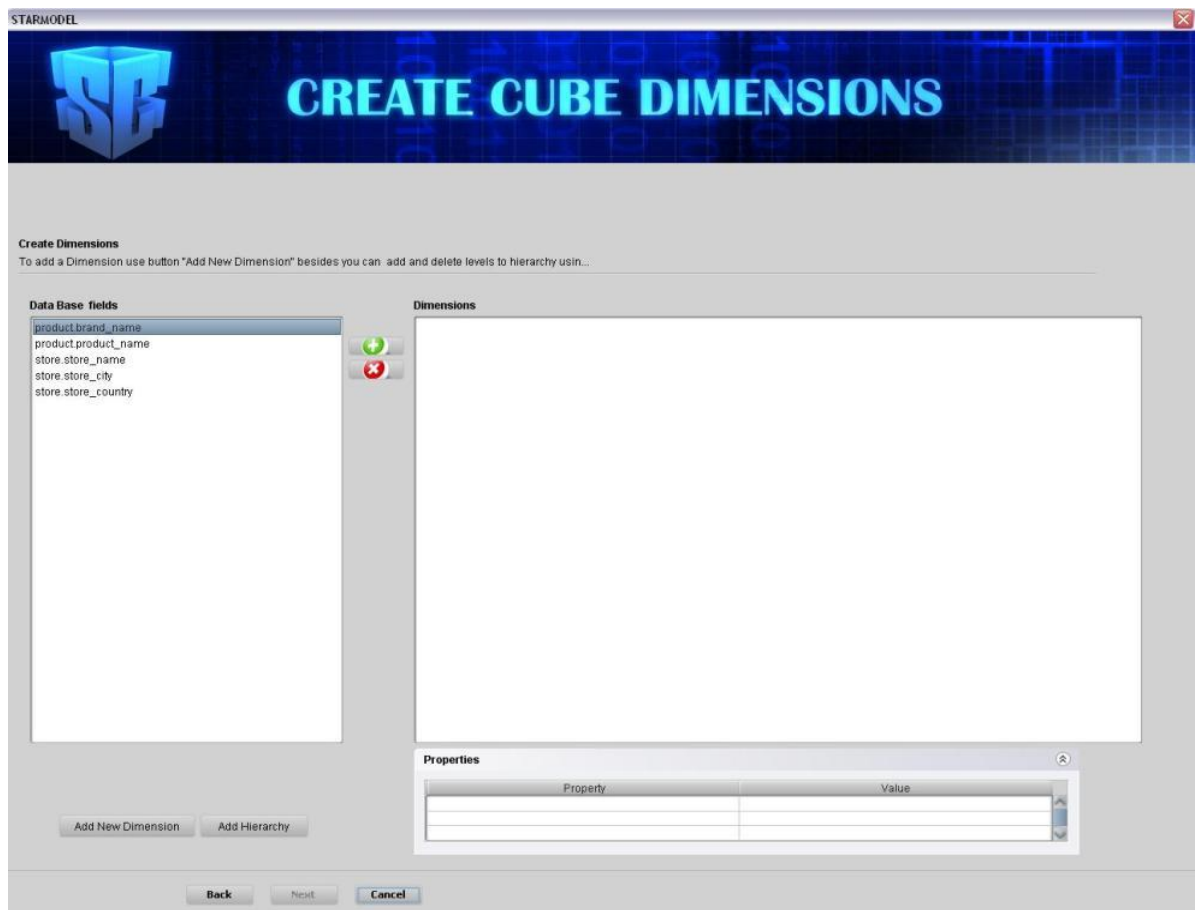
Para remover cualquier Dimensión, Jerarquía o Nivel, selecciónela y haga clic en el botón  (Ver figura 16).

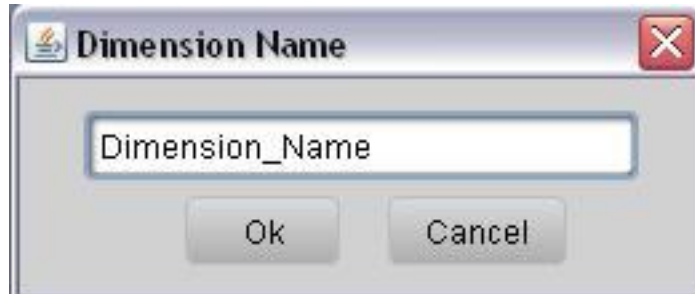
Figura 16. Creación de dimensiones



Cuando se crea una Dimensión aparecerá una ventana “popup” preguntando si se

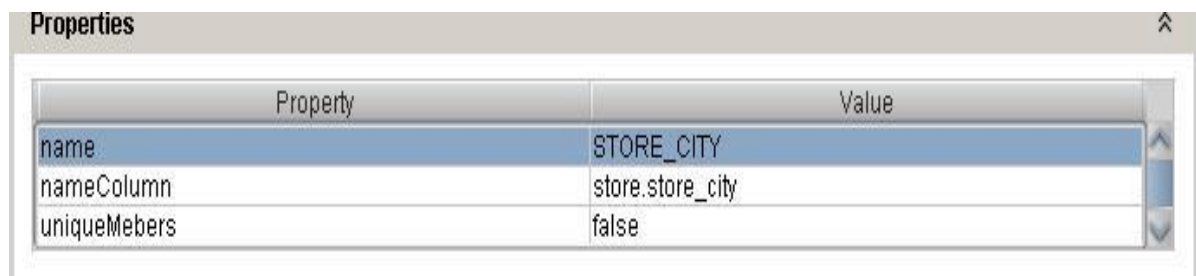
quiere cambiar el nombre de ésta. Se puede escribir cualquier cosa que se desee como nombre de la Dimensión, o simplemente dejar el que tiene (Ver figura 17).

Figura 17. Nombre de la dimensión



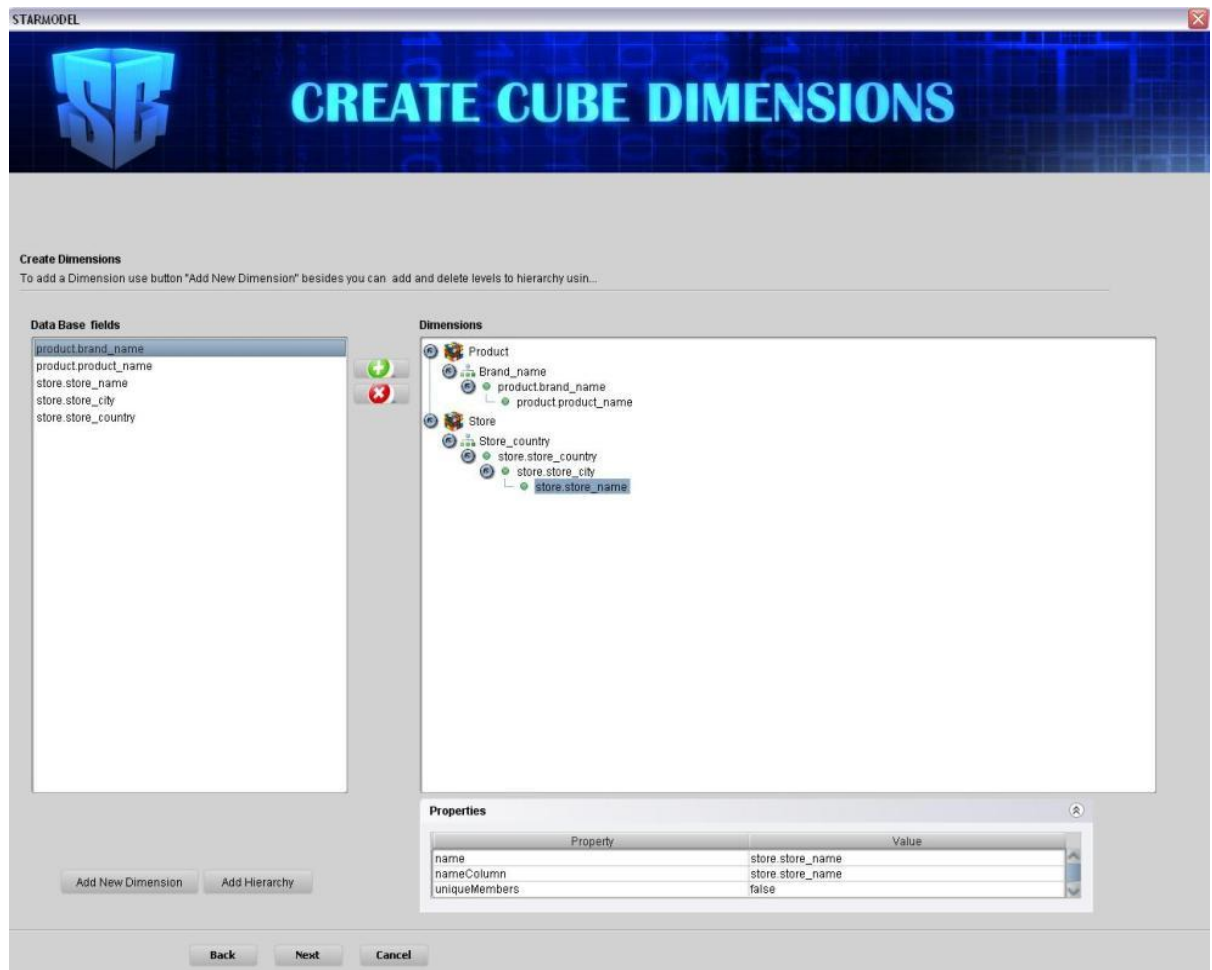
Además, cuando se añade una Dimensión, Jerarquía o Nivel, se puede personalizar sus características en el panel "Properties". En este panel se muestran las propiedades de la Jerarquía/Nivel y sus valores respectivos, los cuales se pueden editar (nombre) o leer información relevante de éste (Nombre de la columna de la BD) (Ver figura 18).

Figura 18. Características de la dimensión, jerarquía o nivel



Cuando se tenga creadas todas las Dimensiones con sus Jerarquías y Niveles, se tendrá algo como lo que se muestra en la figura 19.

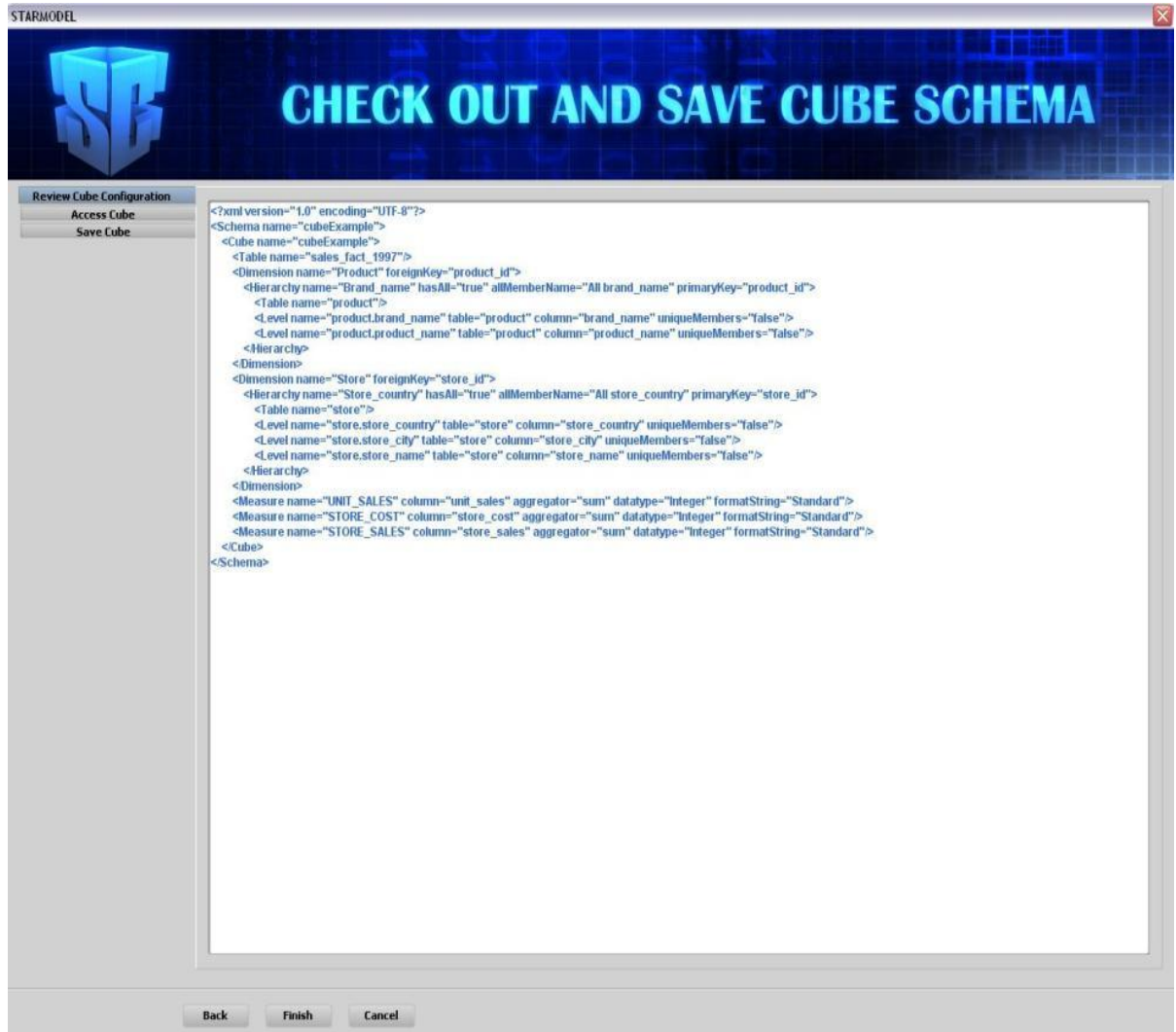
Figura 19. Estructura completa de las dimensiones



### Revisión del esquema del cubo (XML)

Cuando se haya finalizado la creación del cubo, se podrá revisar la configuración de éste (archivo XML), haciendo clic en la pestaña "Review Cube Configuration" (Ver figura 20).

Figura 20. Revisión del esquema del cubo (XML)



## Garantizar el acceso a los usuarios

STARCUBE, provee un control de acceso a los Cubos de datos para los usuarios, permitiendo seleccionar cuáles usuarios pueden trabajar con sus Cubos. Por defecto el modo de acceso es "Public", esto significa que su Cubo será visto por todos los usuarios de STARCUBE. Si se selecciona el modo de acceso "Private" se puede escoger cuáles usuarios tendrán su Cubo en sus listas de trabajo (por defecto el usuario que crea el esquema es seleccionado automáticamente) (Ver figura 21).

Figura 21. Revisión del esquema del Cubo (XML)

Review Cube Configuration

Access Cube

Save Cube

Check: Public or Private Access Cube, default Public.

Private  Public

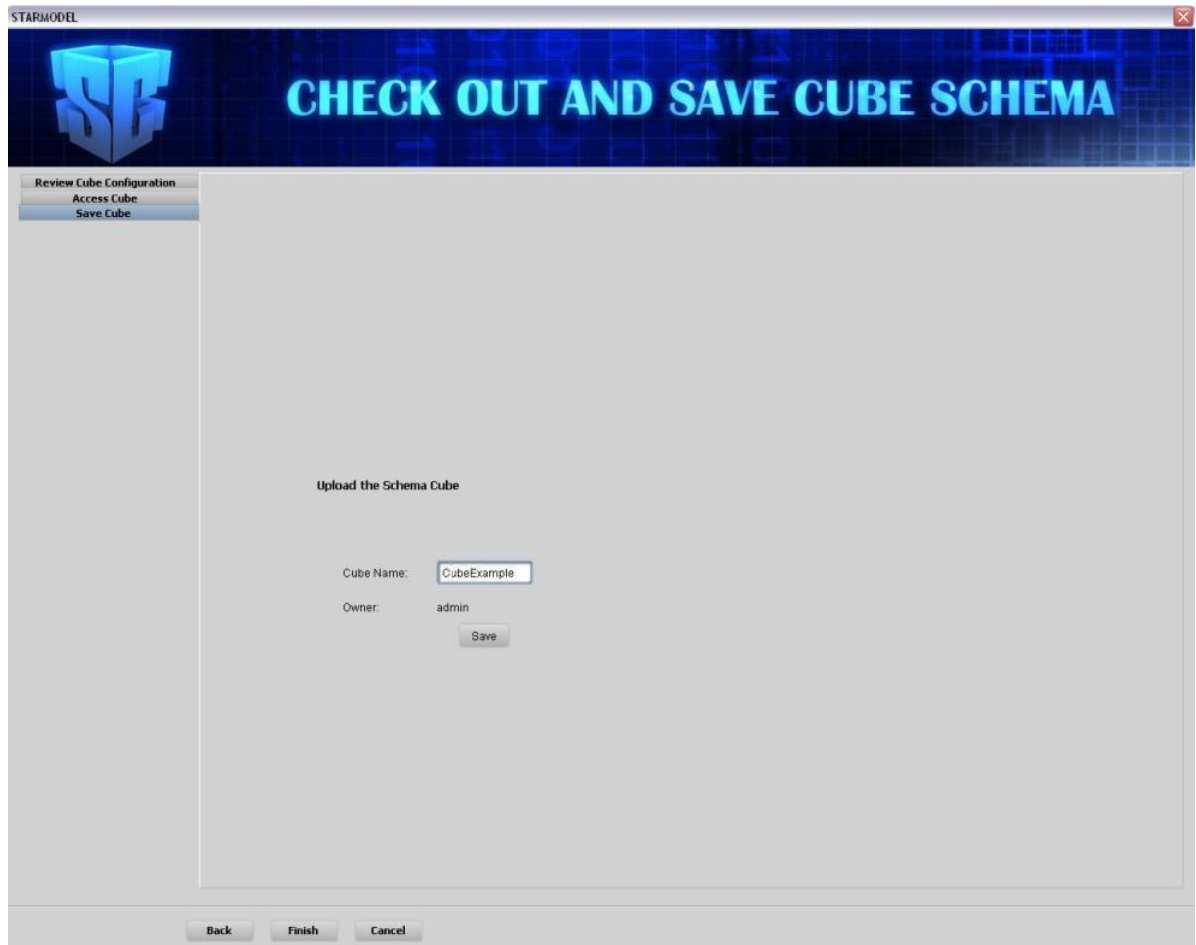
Usuario (Name)	Allow
----------------	-------

### Almacenamiento del Cubo

Finalmente, cuando ud. sabe que su esquema está completo y bien hecho, puede guardarlo. Los cubos almacenados van directamente a la base de datos de PostgreSQL, por lo tanto no es necesario escoger una ruta del sistema (Ver figura 22).

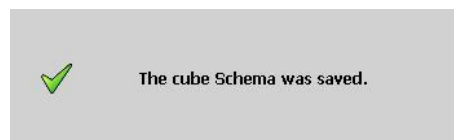


Figura 22. Almacenamiento de un cubo




Cuando se presiona el botón “save” y el Esquema del Cubo está bien hecho y el nombre de éste no se repite, aparecerá un mensaje confirmando que el Esquema fue guardado. De otra forma, aparecerá un mensaje de error (Ver figura 23).

Figura 23. Mensaje de confirmación del guardado de un Cubo



### Abriendo y eliminando cubos de datos

Para abrir o eliminar un Cubo se debe ir al panel “Open Cube” ubicado al lado izquierdo de la ventana, o presionar el botón . En este panel están listados todos los cubos con los que ud. puede trabajar. En la lista desplegable hay dos opciones que puede seleccionar: “My Cubes” donde se encuentran todos los cubos que ud. ha creado y “Shared Cubes” donde están los cubos públicos, compartidos por otros usuarios.


Para abrir cualquier Cubo, tan sólo selecciónelo y presione el botón “Open Cube”.

Para eliminar cualquier Cubo, selecciónelo y presione el botón “Delete Cube” (Ver figura 24).

Figura 24. Navegador de cubos



### Navegando el cubo seleccionado

Para navegar un Cubo OLAP abra el panel “Navigator” en la parte izquierda de la ventana o presione el botón  de la barra de herramientas. En este panel, la información de un Cubo de datos es representada en una estructura de árbol,

mostrando el nombre del Cubo, las Medidas (incluidas en una sola Jerarquía) y las diferentes Dimensiones con sus Jerarquías (Ver figura 25).

Figura 25. Navegación de un cubo



### **Arrastrando y soltando las jerarquías seleccionadas en los campos “ON ROWS” o “ON COLUMNS”**

Para seleccionar qué Dimensiones tendría su análisis, haga clic sobre cualquier Jerarquía, arrástrela desde el árbol y suéltela sobre cualquiera de los dos campos: “ON ROWS” o “ON COLUMNS”

NOTA: Únicamente se puede colocar la Jerarquía de Medidas en uno de los dos lados, para evitar confusiones en el análisis y evitar redundancias.




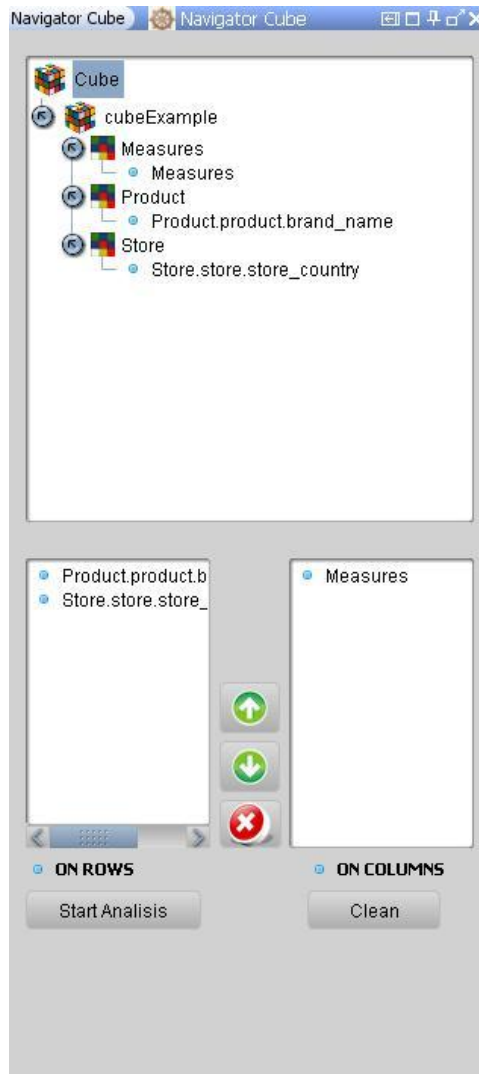
Una vez se hayan escogido las Jerarquías de las Dimensiones y Medidas pueden ser reordenadas en cualquiera de las dos listas, haciendo clic en los botones  ó , o simplemente removerlas usando el botón  (Ver figura 26).

Figura 26. Construcción del análisis



Finalmente, para iniciar el análisis presione el botón “Start Analysis”, pero si se quiere empezar de nuevo, presione el botón “Clean” (quitará todas las Jerarquía presentes).

### **Navegando y manipulando la tabla OLAP y el gráfico**

Cuando se ha comenzado el análisis dos componentes se muestran: La Tabla OLAP y el Gráfico.

En la Tabla OLAP se encontrará toda la información relacionada con su Cubo de Datos representado como una tabla dinámica con estructura interna tipo árbol. En este objeto ud. puede hacer todo tipo de navegaciones como: Drill Down, Roll Up, Pivot y la eliminación de miembros (Ver figura 27).

Figura 27. Tabla OLAP

		Measures	Measures	Measures
Product	Store	UNIT_SALES	STORE_SALES	STORE_COST
+ All product.brand_name	+ All store.store_country	86,837	565,238	225,627

### Operación drill down

Para hacer la operación "Drill Down" haga clic en un miembro que tenga el símbolo "+" en su etiqueta. Esto modificará la Tabla OLAP mostrando los hijos que contiene el miembro clicado. Automáticamente el símbolo del miembro será cambiado por el símbolo "-".

Los miembros que tengan el símbolo "●" en su etiqueta son los últimos miembros de la estructura (hojas), en los cuales únicamente no podrá realizarse las operaciones Drill Down o Roll up (Ver figura 28).

Figura 28. Realización de la operación drill down / roll up en los miembros

Product	Store	Measures	Measures	Measures
		UNIT_SALES	STORE_SALES	STORE_COST
+ All product.brand_name	- All store.store_country	86,837	565,238	225,627
	+ Canada			
	+ Mexico			
	- USA	86,837	565,238	225,627
	+ Alameda			
	+ Bellingham	1,380	4,739	1,897
	+ Beverly Hills	6,815	45,750	18,266
	+ Bremerton	7,876	52,896	21,122
	+ Los Angeles	8,207	54,545	21,772
	- Portland	8,264	55,059	21,949
	+ Store 11	8,264	55,059	21,949
	+ Salem	13,347	87,218	34,824
	+ San Diego	8,095	54,431	21,714
	+ San Francisco	1,325	4,441	1,779
	+ Seattle	7,956	52,644	20,957
	+ Spokane	7,397	49,634	19,795
	+ Tacoma	11,184	74,844	29,959
	- Walla Walla	1,339	4,706	1,880
	+ Store 22	1,339	4,706	1,880
	- Yakima	3,652	24,329	9,714
	+ Store 23	3,652	24,329	9,714

### Operación roll up

Esta operación es la recíproca a “Drill Down”, pero es muy similar, excepto que en este caso se debe hacer clic en los Miembros que tengan el símbolo “-” en su etiqueta. El resultado de esta operación es subir un nivel en la Jerarquía. (Ver figura 28).

### Operación pivot


Para hacer la operación “Pivot” tan sólo haga clic en la esquina superior izquierda de la Tabla OLAP o presione el botón  en la barra de herramientas. Esto automáticamente intercambiará los ejes de la Tabla (Ver figura 29).

Figura 29. Realización de la operación pivot en la tabla OLAP

	Product	
	+ All product.brand_name	
	Store	
Measures	+ All store.store_country	
UNIT_SALES	86,837	
STORE_SALES	565,238	
STORE_COST	225,627	

### Manipulación de los miembros de la tabla

- **Eliminar un miembro:** Para eliminar un solo Miembro, selecciónelo haciendo clic derecho, presione el botón central del ratón (MouseWheelButton) para desplegar el menú "Delete Member" y posteriormente haga clic en esa opción (Ver figura 30).

Figura 30. Operación de eliminación de un miembro en la tabla OLAP

The screenshot shows a window titled 'Starcube Table' containing a table with three columns: 'Product', 'Store', and three 'Measures' columns: 'UNIT\_SALES', 'STORE\_COST', and 'STORE\_SALES'. The table is organized into a hierarchy. The 'Product' column has a main level 'All product.brand\_name' and a sub-level 'All store.store\_country'. The 'Store' column lists various locations. A 'Delete Member' button is overlaid on the row for 'Store 11'.

Product	Store	Measures	Measures	Measures
		UNIT_SALES	STORE_COST	STORE_SALES
All product.brand_name	- All store.store_country	266,773	225,627	565,238
	+ Canada			
	+ Mexico			
	- USA	266,773	225,627	565,238
	+ Alameda			
	+ Bellingham	2,237	1,897	4,739
	+ Beverly Hills	21,333	18,266	45,750
	+ Bremerton	24,576	21,122	52,896
	+ Los Angeles	25,663	21,772	54,545
	- Portland	26,079	21,949	55,059
	+ Store 11	26,079	21,949	55,059
	+ Salem	41,580	34,824	87,218
	+ San Diego	25,635	21,714	54,431
	+ San Francisco	2,117	1,779	4,441
	+ Seattle	25,011	20,957	52,644
	+ Spokane	23,591	19,795	49,634
	+ Tacoma	35,257	29,959	74,844
	- Walla Walla	2,203	1,880	4,706
	+ Store 22	2,203	1,880	4,706
	- Yakima	11,491	9,714	24,329
+ Store 23	11,491	9,714	24,329	

**Resultado:** El resultado de esta operación es la eliminación del Miembro seleccionado (Ver figura 31).



Figura 31. Resultado de la eliminación de un miembro en la tabla OLAP

Product		Measures	Measures	Measures
Store		UNIT_SALES	STORE_COST	STORE_SALES
-	All store.store_country	266,773	225,627	565,238
+	Canada			
+	Mexico			
-	USA	266,773	225,627	565,238
+	Alameda			
+	Bellingham	2,237	1,897	4,739
+	Beverly Hills	21,333	18,266	45,750
+	Bremerton	24,576	21,122	52,896
+	Los Angeles	25,663	21,772	54,545
+	Store 11	26,079	21,949	55,059
+	Salem	41,580	34,824	87,218
+	San Diego	25,635	21,714	54,431
+	San Francisco	2,117	1,779	4,441
+	Seattle	25,011	20,957	52,644
+	Spokane	23,591	19,795	49,634
+	Tacoma	35,257	29,959	74,844
-	Walla Walla	2,203	1,880	4,706
+	Store 22	2,203	1,880	4,706
-	Yakima	11,491	9,714	24,329
+	Store 23	11,491	9,714	24,329

- Eliminar un grupo de miembros:** Para eliminar un grupo de Miembros selecciónelos teniendo presionada la tecla CTRL, cuando se haya realizado la selección, presione el botón central del ratón sobre cualquiera de los miembros (MouseWheelButton) para desplegar el menú "Delete Members Selection" y haga clic en esa opción (Ver figura 32).

Figura 32. Operación de eliminación de un grupo de miembros en la tabla OLAP

Product		Measures	Measures	Measures
Store		UNIT_SALES	STORE_COST	STORE_SALES
+ All product.brand_name	- All store.store_country	266,773	225,627	565,238
	+ Canada			
	+ Mexico			
	- USA	266,773	225,627	565,238
	+ Alameda			
	+ Bellingham	2,237	1,897	4,739
	+ Beverly Hills	21,333	18,266	45,750
	+ Bremerton	24,576	21,122	52,896
	+ Los Angeles	25,663	21,772	54,545
	- Portland		21,949	55,059
	+ Store 11		21,949	55,059
	+ Salem		34,824	87,218
	+ San Diego	25,835	21,714	54,431
	+ San Francisco	2,117	1,779	4,441
	+ Seattle	25,011	20,957	52,644
	+ Spokane	23,591	19,795	49,634
	+ Tacoma	35,257	29,959	74,844
	- Walla Walla	2,203	1,880	4,706
	+ Store 22	2,203	1,880	4,706
	- Yakima	11,491	9,714	24,329
+ Store 23	11,491	9,714	24,329	

**Resultado:** El resultado de esta operación, es la eliminación de los miembros seleccionados (Ver figura 33).

Figura 33. Resultado de la eliminación de un grupo de miembros en la tabla OLAP

Starcube Table		Measures	Measures	Measures
Product	Store	UNIT_SALES	STORE_COST	STORE_SALES
+ All product.brand_name	- All store.store_country	266,773	225,627	565,238
	+ Canada			
	+ Mexico			
	- USA	266,773	225,627	565,238
	+ Alameda			
	+ Bellingham	2,237	1,897	4,739
	+ Beverly Hills	21,333	18,266	45,750
	+ Bremerton	24,576	21,122	52,896
	+ Los Angeles	25,663	21,772	54,545
	+ Store 11	26,079	21,949	55,059
	+ Salem	41,580	34,824	87,218
	+ San Diego	25,635	21,714	54,431
	+ San Francisco	2,117	1,779	4,441
	+ Seattle	25,011	20,957	52,644
	+ Spokane	23,591	19,795	49,634
	+ Tacoma	35,257	29,959	74,844
	+ Store 22	2,203	1,880	4,706
	+ Store 23	11,491	9,714	24,329

- Mantener una selección de miembros:** Para mantener un grupo de Miembros, selecciónelos teniendo presionada la tecla CTRL, cuando se haya realizado la selección, presione el botón central del ratón sobre cualquiera de los miembros (MouseWheelButton) para desplegar el menú "Hold Members Selection" y haga clic en esa opción (Ver figura 34).

Figura 34. Resultado de mantener un grupo de miembros en la tabla OLAP


Product		Measures	Measures	Measures
Store		UNIT_SALES	STORE_COST	STORE_SALES
-	All store.store_country	266,773	225,627	565,238
+	Canada			
+	Mexico			
-	USA	266,773	225,627	565,238
+	Alameda			
+	Bellingham	2,237	1,897	4,739
+	Beverly Hills	21,333	18,266	45,750
+	Bremerton	24,576	21,122	52,896
+	Los Angeles	25,663	21,772	54,545
-	Portland	26,079	21,949	55,059
+	Store 11		21,949	55,059
+	Salem		34,824	87,218
+	San Diego	23,833	21,714	54,431
+	San Francisco	2,117	1,779	4,441
+	Seattle	25,011	20,957	52,644
+	Spokane	23,591	19,795	49,634
+	Tacoma	35,257	29,959	74,844
-	Walla Walla	2,203	1,880	4,706
+	Store 22	2,203	1,880	4,706
-	Yakima	11,491	9,714	24,329
+	Store 23	11,491	9,714	24,329

**Resultado:** El resultado de esta operación es la conservación de los miembros seleccionados (Ver figura 35).

Figura 35. Resultado de mantener de un grupo de Miembros en la Tabla OLAP

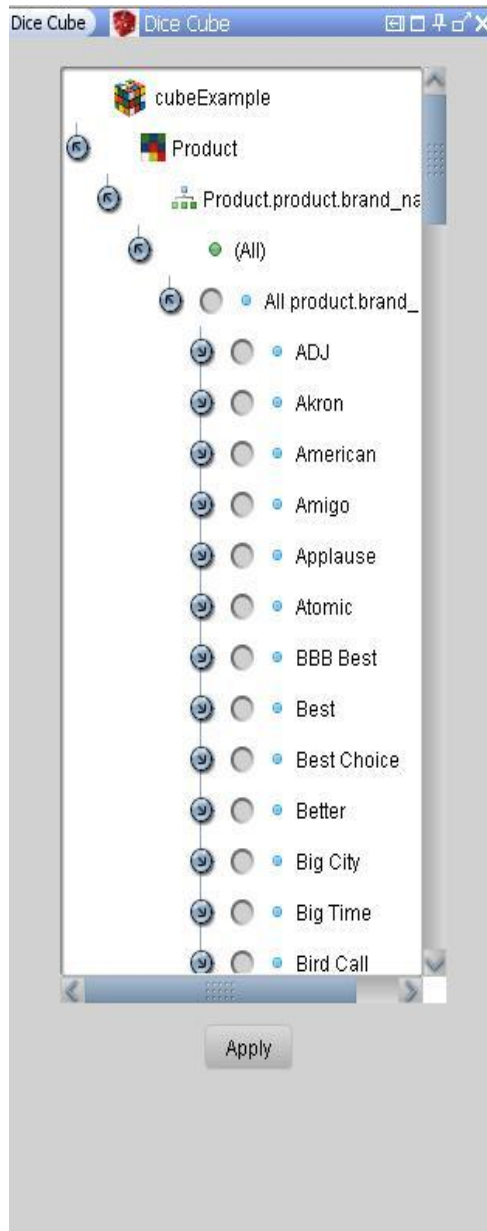
Product		Measures	Measures	Measures
Store		UNIT_SALES	STORE_COST	STORE_SALES
-	All store.store_country	266,773	225,627	565,238
+	Portland	26,079	21,949	55,059
+	Walla Walla	2,203	1,880	4,706
+	Yakima	11,491	9,714	24,329

**Operación dice:**


Para hacer una operación “Dice”, abra el panel “Dice Cube” en la parte derecha de la ventana o haga clic en el botón , y luego seleccione cuál Miembro será la restricción para su consulta OLAP.

NOTA: Para hacer la operación “Dice”, se debe tener al menos una Dimensión fuera del análisis actual, es decir, que no se encuentre en la Tabla OLAP (Ver figura 36).

Figura 36. Navegador para la operación dice

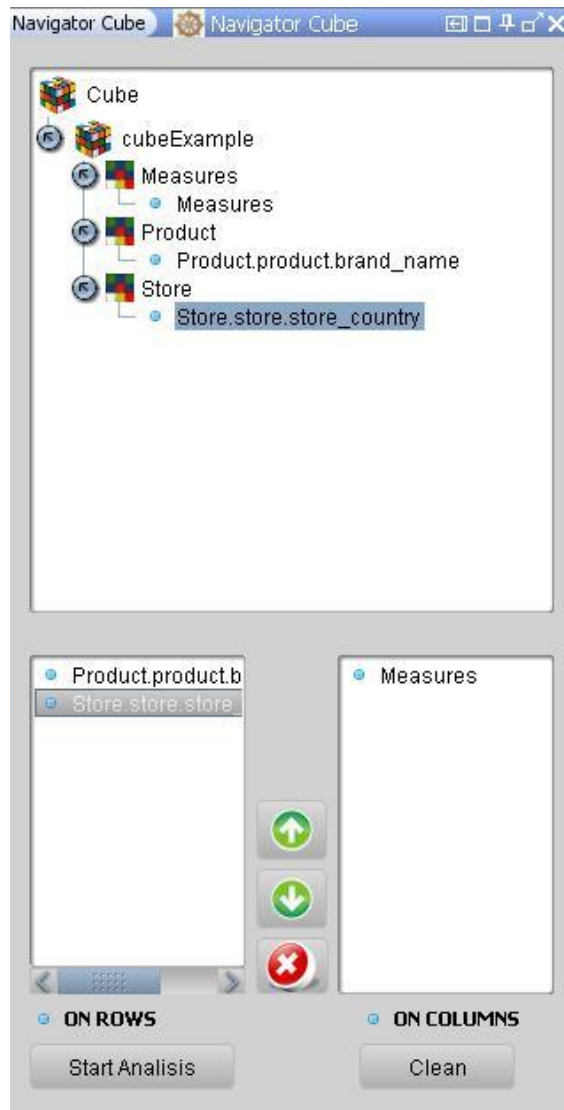


## Operación slice


Para llevar a cabo la operación "Slice" use el panel "Navigator" (parte izquierda de la ventana o clic en el botón  de la barra de herramientas), el cual permite agregar, remover y reordenar cualquier Jerarquía de las listas "ON ROWS" o "ON COLUMNS".

Escoja las nuevas Jerarquías que le gustaría que su análisis, o elimine las que no quiere dentro de su análisis, y luego presione el botón "Start Analysis" nuevamente. Esto reestructurará la tabla automáticamente (Ver figura 37).

Figura 37. Operación slice utilizando el panel navigator

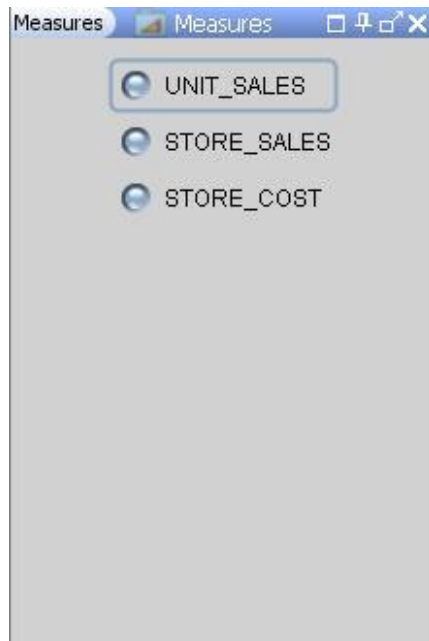


## Añadir o eliminar medidas


Para añadir o eliminar una medida de la tabla OLAP, diríjase al panel “Measures” en la parte derecha de la ventana o haga clic en el botón  en la barra de herramientas y seleccione cuáles medidas quiere o no, mostrar en la tabla, haciendo clic en las casillas de verificación (Ver figura 38).

NOTA: Al hacer esta operación quedará al menos una medida seleccionada.

Figura 38. Manipulación de las medidas del cubo

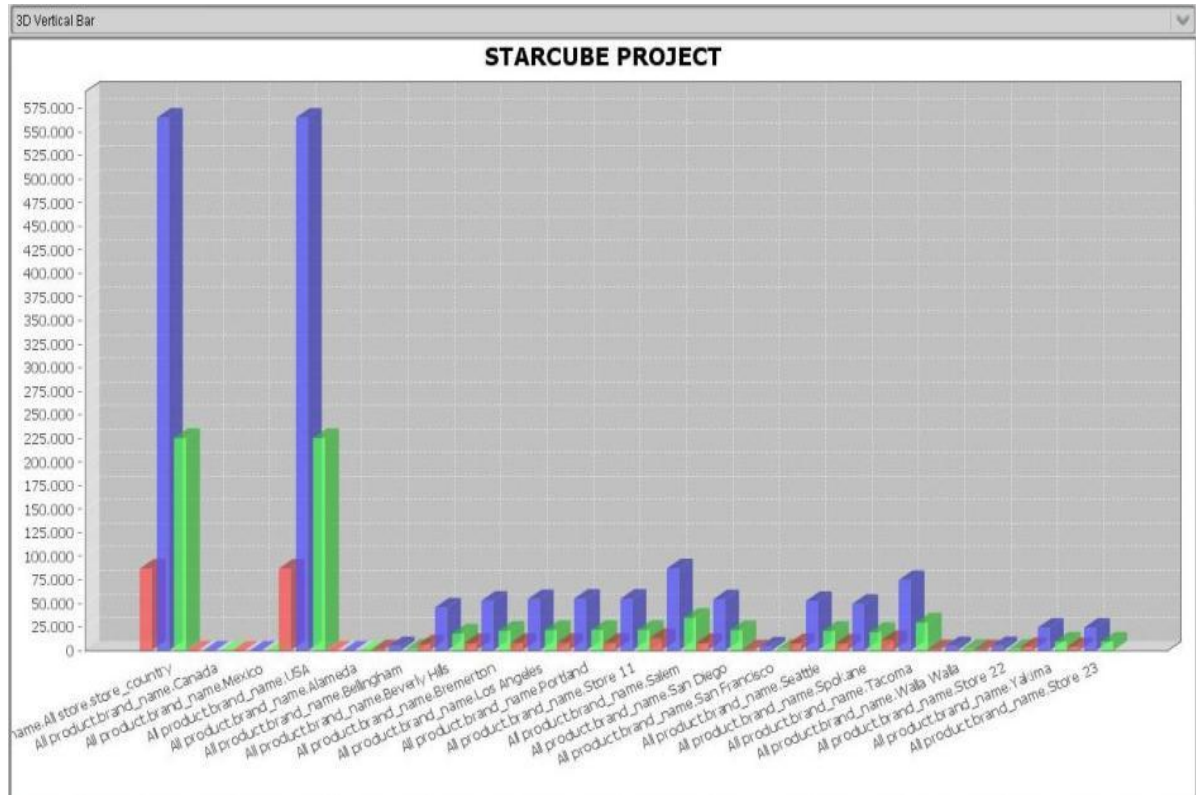


## Gráfica

Un componente importante de un análisis OLAP es el objeto Gráfico. En este útil elemento se muestra toda la información contenida en la Tabla OLAP, haciendo más fácil el análisis y la toma de decisiones. Para observar u ocultar el Gráfico presione el botón . En este componente Ud. será capaz de escoger el tipo de gráfica haciendo clic en la lista de la parte superior del panel.

Adicionalmente, se puede personalizar y guardar la gráfica, haciendo clic derecho en el componente y seleccionando cualquiera de las opciones mostradas (Ver figura 39).




Figura 39. Gráfica





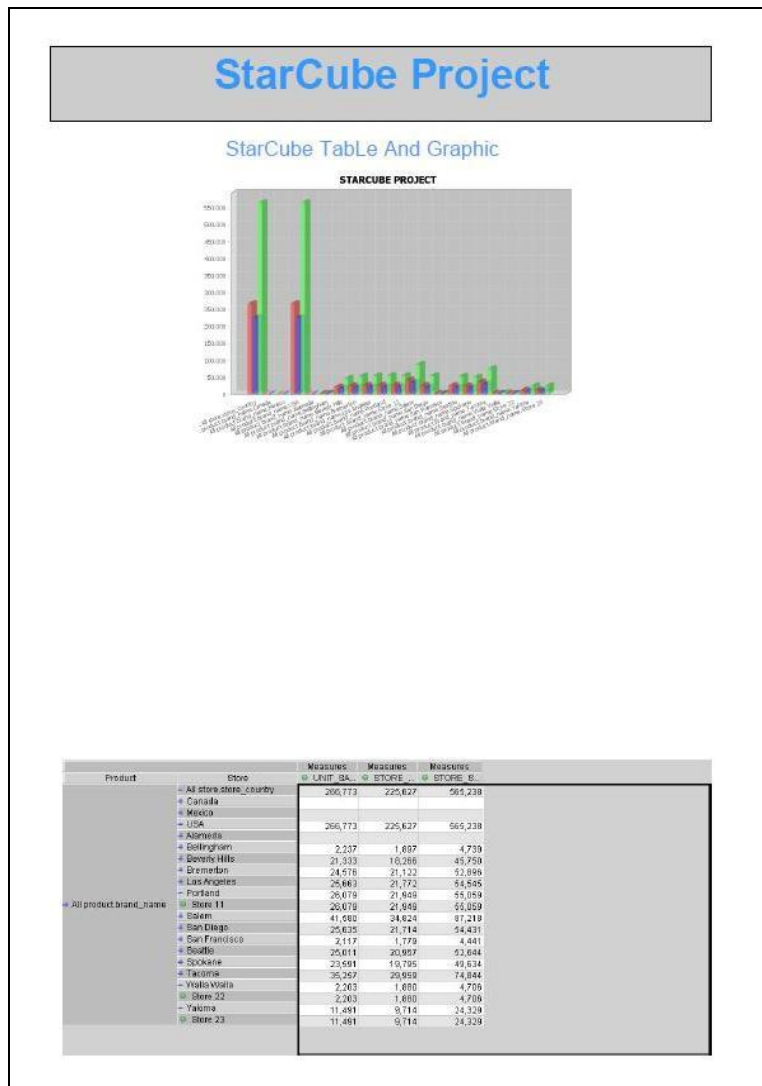
## REPORTES

### Exportar datos

STARCUBE permite guardar un análisis en diferentes tipos de reportes, como lo son: PDF, HTML y EXCEL. Para exportar un análisis use los botones ,  o  para generar el respectivo reporte.

Para los reportes en formato pdf o HTML la información tiene dos componentes: Una imagen de la tabla OLAP y una imagen de la gráfica (Ver figura 40).

Figura 40. Reporte en formato pdf y HTML



Para el reporte en formato Excel la información es enviada en tres maneras: Datos puros (en celdas de Excel), una imagen de la tabla OLAP y una imagen de la gráfica (Ver figuras 41 a 43).

**Datos puros (en celdas de EXCEL):**

Figura 41. Reporte en formato EXCEL (datos puros)

	A	B	C	D	E	F
1	Product	Store	UNIT_SALES	STORE_COST	STORE_SALES	
2	All product.brand_name	All store.store_country	266,773	225,627	565,238	
3	All product.brand_name	Canada				
4	All product.brand_name	Mexico				
5	All product.brand_name	USA	266,773	225,627	565,238	
6	All product.brand_name	Alameda				
7	All product.brand_name	Bellingham	2,237	1,897	4,739	
8	All product.brand_name	Beverly Hills	21,333	18,266	45,750	
9	All product.brand_name	Bremerton	24,576	21,122	52,896	
10	All product.brand_name	Los Angeles	25,663	21,772	54,545	
11	All product.brand_name	Portland	26,079	21,949	55,059	
12	All product.brand_name	Store 11	26,079	21,949	55,059	
13	All product.brand_name	Salem	41,580	34,824	87,218	
14	All product.brand_name	San Diego	25,635	21,714	54,431	
15	All product.brand_name	San Francisco	2,117	1,779	4,441	
16	All product.brand_name	Seattle	25,011	20,957	52,644	
17	All product.brand_name	Spokane	23,591	19,795	49,634	
18	All product.brand_name	Tacoma	35,257	29,959	74,844	
19	All product.brand_name	Walla Walla	2,203	1,880	4,706	
20	All product.brand_name	Store 22	2,203	1,880	4,706	
21	All product.brand_name	Yakima	11,491	9,714	24,329	
22	All product.brand_name	Store 23	11,491	9,714	24,329	
23						

**Imagen de la tabla:**

Figura 42. Reporte en formato EXCEL (imagen tabla OLAP)

The screenshot shows the Microsoft Excel interface with the following data table:

	Product	Store	Measures UNIT_SA...	Measures STORE...	Measures STORE_S...
2		All store.store_country	266,773	225,627	565,238
3		+ Canada			
4		+ Mexico			
5		- USA	266,773	225,627	565,238
6		+ Alameda			
7		+ Bellingham	2,237	1,897	4,739
8		+ Beverly Hills	21,333	18,266	45,750
9		+ Bremerton	24,576	21,122	52,896
10		+ Los Angeles	25,863	21,772	54,545
11		- Portland	26,079	21,949	55,059
12		+ Store 11	26,079	21,949	55,059
13		+ Salem	41,580	34,824	87,218
14		+ San Diego	25,635	21,714	54,431
15		+ San Francisco	2,117	1,779	4,441
16		+ Seattle	25,011	20,957	52,644
17		+ Spokane	23,591	19,795	49,634
18		+ Tacoma	35,257	29,958	74,844
19		- Walla Walla	2,203	1,880	4,706
20		+ Store 22	2,203	1,880	4,706
21		- Yakima	11,491	9,714	24,329
22		+ Store 23	11,491	9,714	24,329

Imagen de la gráfica:

Figura 43. Reporte en formato EXCEL (imagen de la gráfica)

