Tesis de Maestría.

CMDM: Un Modelo Conceptual para la Especificación de Bases Multidimensionales.

Fernando Carpani.

Agosto 2000

Tutor: Dr. Raúl Ruggia.

Instituto de Computación - Facultad de Ingeniería.
Universidad de la República.
Pedeciba Informática.

Resumen.

En los últimos años, el área de Data Warehouse y Aplicaciones Olap ha tenido un desarrollo importante. En este tipo de aplicaciones, se construye una una base de datos con visión multidimensional de la realidad.

A pesar del amplio desarrollo del área, aún no hay mecanismos de especificación de este tipo de sistemas que permita tener en cuenta la mayor parte de los detalles relevantes de una determinada porción de la realidad. Las especificaciones relacionales ([Kim96]) dejan de lado algunos aspectos multidimensionales como la dimensionalidad genérica ([Cod93]). Las especificaciones multidimensionales que manejan la dimensionalidad genérica, contemplan los aspectos de carga y/o limpieza de los datos desde datos de un nivel inferior de una forma externa al modelo o directamente, no los contemplan. Pocas propuestas son capaces de permitir la especificación de qué manipulaciones están autorizadas y cuales no sobre determinado aspecto de la realidad.

En este trabajo, se presenta un modelo que permite la especificación detallada de una base de datos multidimensional. Esta especificación se construye medienta un lenguaje gráfico que permite describir las estructuras de datos y algunas restricciones de integridad, y un lenguaje de restricciones de integridad que permite dar una descripción precisa de las relaciones entre los datos.

Tabla de Contenido.

Figuras.

Definiciones.

Capítulo 1. Introducción.

1.1. Motivación.

Las empresas, acumulan datos de su gestión con fines de registro (contabilidad, stock, facturación, etc.).

Por otro lado, los encargados de definir las políticas estratégicas de las empresas deben realizar análisis de estos datos de gestión. Estos análisis son presentados bajo la forma de resúmenes, como por ejemplo, promedios o totalizaciones de algunos indicadores en función de otras variables.

Este tipo de manipulación de los datos es típicamente exploratorio, es decir, en función de los resultados de un reporte se decide cuál es el siguiente que se debe realizar. Esto implica que los usuarios que deben realizar estos análisis deben acceder ellos mismos a grandes cantidades de datos, posiblemente ya resumidos, y de forma eficiente.

Esto constituye una clase de sistemas de información conocida como Sistemas de Apoyo a la Toma de Decisiones, actualmente conocidos como Sistemas de *Data Warehousing*.

Desde el punto de vista del software, las herramientas en que se basaban originalmente estos sistemas, eran las planillas de cálculo y las bases de datos. Sin embargo, la utilización directa de estas herramientas presenta ciertos problemas para este tipo de tareas. Las bases de datos relacionales no permiten directamente la presentación de los datos en forma adecuada para las tareas de análisis, además exigen que los usuarios sean expertos en lenguajes de consulta, típicamente SQL. Las planillas de cálculo proveen una representación adecuada y manejable por usuarios relativamente inexpertos, sin embargo presentan grandes problemas de eficiencia, e incluso de funcionamiento, en los casos en que se debe realizar acceso masivo a grandes cantidades de datos.

En la última década han surgido en el mercado un conjunto de productos conocidos como interfases y servidores *OLAP* que montados sobre Data Warehouses, tratan de resolver algunos de los problemas anteriores.

Estos productos surgidos originalmente de la industria, han “reciclado” algunas ideas como la de representar los datos como funciones de varias variables o matrices multidimensionales.

Sin embargo, a pesar de haberse difundido bastante, estos productos no se basan en un modelo de datos bien definido ni ampliamente aceptado sino que existe un conjunto de funcionalidades y operaciones que todos proveen, y otras que son las que diferencia un producto de otro.

Los sistemas de Data Warehousing y basados en herramientas Olap, presentan algunas características propias que hacen que los modelos conceptuales tradicionales como el Modelo Entidad-Relación u otros orientados a objetos no resulten demasiado adecuados. Esto se debe, entre otras características, a que los datos deben ser vistos desde un punto de vista multidimensional.

En los últimos tres años, han surgido varios modelos de diversa índole que permiten la descripción conceptual y/o lógica de los aspectos multidimensionales de una realidad determinada.

Sin embargo, en estos modelos no se han propuesto en forma explícita lenguajes de restricciones que permitan la realización de especificaciones precisas.

Por otro lado, algunas metodologías y modelos asumen que la implementación del Data Warehouse se realizará en un manejador de base de datos relacional. Esta postura deja de lado a los manejadores de bases de datos multidimensionales de lado, sin considerarlos siquiera como una alternativa posible.

1.2. Visión General.

1.2.1. Nociones Básicas de Data Warehousing.

Un *Data Warehouse* es una base de datos con resúmenes precalculados que centraliza todos los datos de la empresa. Esta base de datos se carga a partir de los datos operativos (facturación, stock, etc.) de la empresa, típicamente mediante un mecanismo batch.

Un *Sistema de Data Warehousing* es un Sistema de Información Orientado a la Toma de Decisiones y se basa en la explotación de un Data Warehouse.

En la Fig. 1 se puede ver la arquitectura típica de un sistema de Data Warehousing.

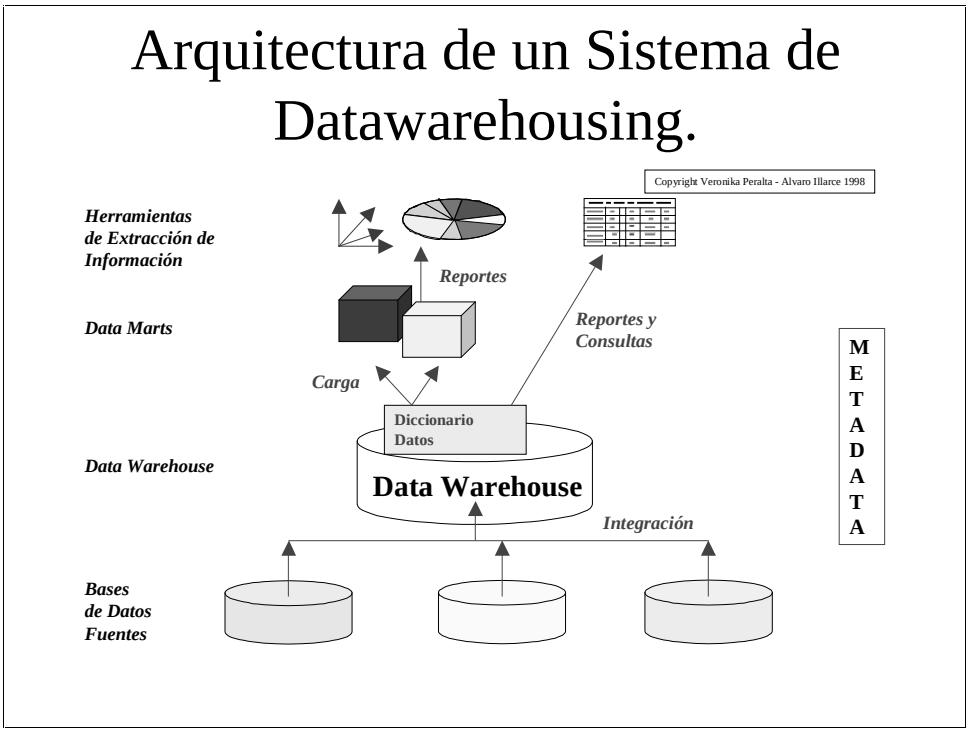

Fig. 1. Arquitectura de Referencia de un Sistema de DW.

El esquema del Data Warehouse se construye sobre la base de un proceso de integración aplicado sobre cierto conjunto de bases de datos fuentes. Se carga con datos que surgen de un proceso conocido como extracción y limpieza, en donde sólo se cargan los datos y se construyen los resúmenes que sean relevantes para las tareas de toma de decisiones.

A partir del Data Warehouse, se construyen Data Marts que son pequeños Data Warehouses orientados a áreas específicas de la empresa. Podría existir uno para la parte de ventas, otro para la parte de manufactura, etc.

Tanto a partir de los Data Marts como del Data Warehouse, se construyen reportes y gráficos en base a herramientas de extracción de información. Estas herramientas de extracción de información pueden ser de diversos tipos, pero típicamente permiten manipular la información de forma similar a una planilla de cálculo.

El problema que se pretende resolver en esta tesis es el de la especificación del Data Warehouse y los Data Marts necesarios, teniendo en cuenta que los datos deben ser vistos de forma multidimensional, es decir, como matrices n-dimensionales.

1.3. Aportes y Limitaciones.

Hasta el momento, los modelos encontrados no proponen en forma explícita un lenguaje de restricciones de integridad. Sólo proponen estructuras de datos y en algunos casos mecanismos de consultas sobre esas estructuras de datos.

Tradicionalmente, las estructuras que se utilizan para representar diferentes formas de análisis (cubos) se definen individualmente, una por una. Esto tiene la desventaja de no se muestran todas las opciones de análisis disponibles para quien interpreta los datos.

En CMDM, se proponen estructuras de datos y un mecanismo de especificación de restricciones de integridad. Con este mecanismo se pueden especificar conjuntos de cubos con características comunes, permitiendo la especificación de un conjunto mayor de opciones de análisis que simplemente las que se den una a una.

Dado que el modelo está orientado al diseño conceptual, no se propone en forma explícita un lenguaje de consulta.

Además, se presenta el diseño y un prototipo de una herramienta CASE orientado al diseño de Data Warehouse.

1.4. Organización del Documento.

En el siguiente capítulo se presentan las bases de los modelos multidimensionales y una pequeña revisión del estado del arte sobre los modelos conceptuales para bases multidimensionales. En el Anexo I se presenta un reporte técnico que presenta con mayor detalle los diferentes modelos conceptuales y lógicos encontrados en la bibliografía hasta el momento.

En el capítulo 3 se propone una primera versión del modelo, presentando primero los fundamentos del modelo y luego las estructuras de datos a través de un ejemplo. Finalmente se presenta una formalización de las estructuras de datos.

En el capítulo 4 se presenta un lenguaje de restricciones aplicándolo al ejemplo anterior que luego se formaliza. Se propone una nueva versión del modelo capaz de soportar las restricciones de integridad.

En el capítulo 5 se presenta un prototipo de una herramienta CASE para el diseño de bases multidimensionales usando CMDM.

Capítulo 2. Estado del Arte.

En este capítulo, se presenta una breve descripción de otros trabajos en modelado conceptual y lógico de bases de datos multidimensionales. Para facilitar la comprensión de dichos trabajos y unificar terminología, previamente se presentará una introducción a las estructuras y operaciones de los modelos multidimensionales.

1.1. Introducción a los Modelos Multidimensionales.

2.1.1. Estructuras Básicas: Cubos, Dimensiones y Medidas.

En una base de datos multidimensional, la información se representa como matrices multidimensionales, cuadros de múltiples entradas o funciones de varias variables sobre conjuntos finitos. Cada una de estas matrices se denomina ***Cubo***.

El esquema de un cubo queda determinado dando a conocer sus ejes con sus respectivas estructuras y la estructura de los datos que se presentan en cada celda de la matriz. Se asume que los datos en todas las celdas son uniformes, es decir, todas las posiciones de la matriz tienen datos con igual estructura.

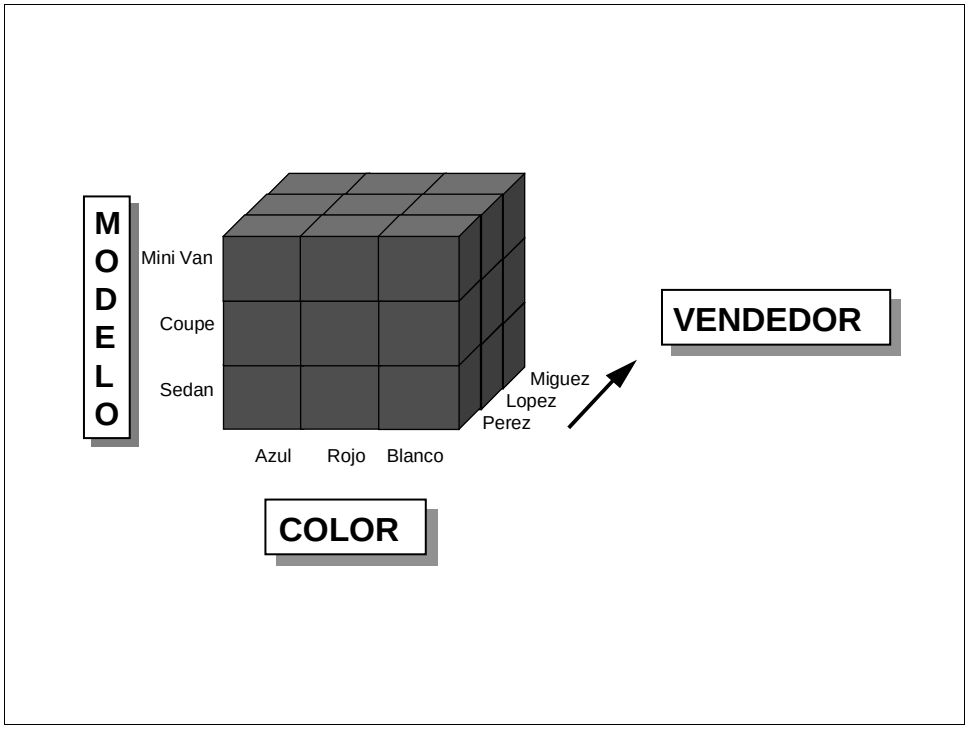

Fig. 2. La estructura básica en el modelo multidimensional: el Cubo.

Una instancia de un cubo, queda determinada por un conjunto de datos para cada eje y un conjunto de datos para la matriz. (Fig. 3)

A los ejes se les llama ***Dimensiones*** y al dato que se presenta en la matriz, se le llama ***Medida***. A los elementos del producto cartesiano de los ejes (dimensiones) se le llama ***Coordenadas***. La matriz definida, puede ser dispersa. (Es una función parcial).

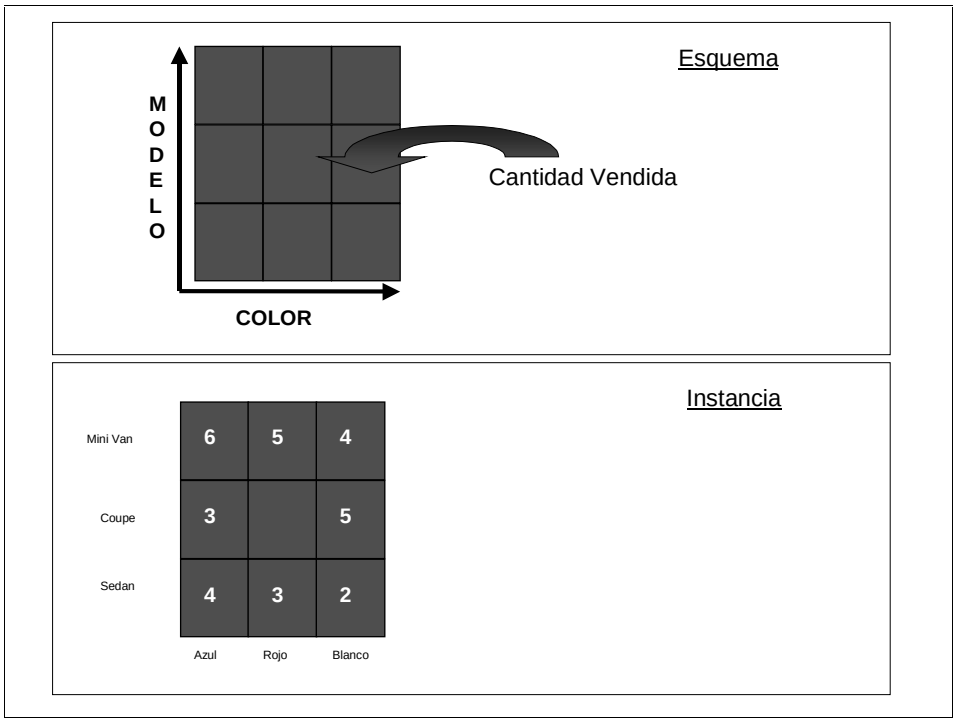

Fig. 3. Esquema e instancia de un cuadro de doble entrada.

2.1.2. Dimensiones con Jerarquías.

Generalmente, las dimensiones se estructuran en jerarquías de agregación. En la Fig. 4 se muestra una instancia de una dimensión con una jerarquía de agregación en donde los vendedores se agrupan en ciudades y las ciudades en regiones.

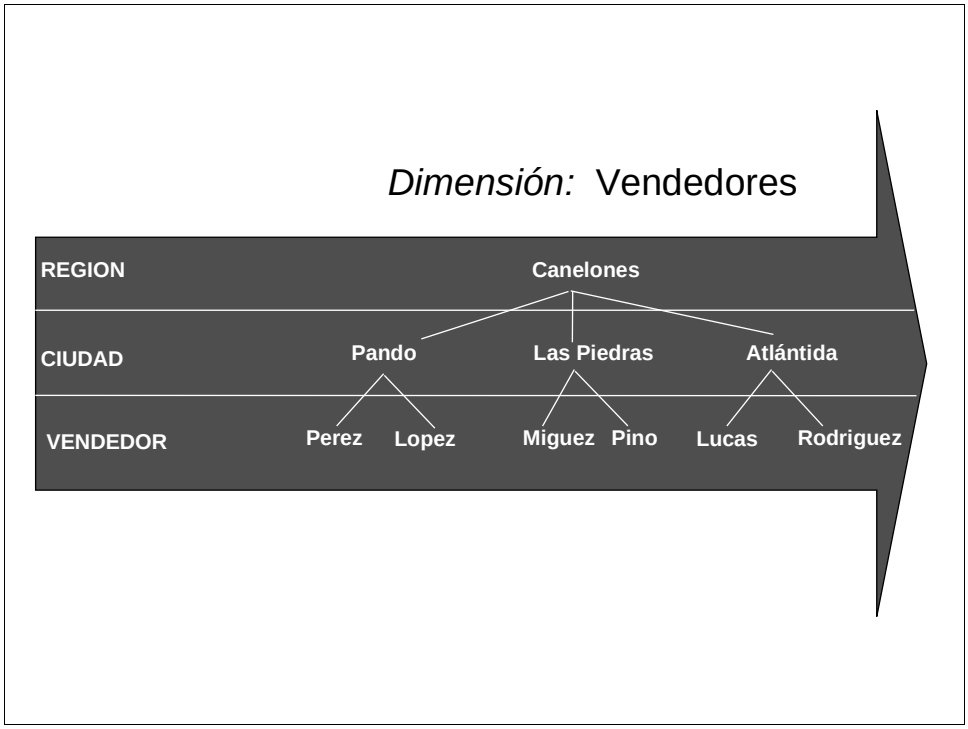

Fig. 4. Dimensión con una Jerarquía.

A cada nivel de una jerarquía se le llama ***Nivel de Agregación*** o simplemente ***Nivel***.

De esta forma, se puede considerar que toda dimensión siempre tiene por lo menos una jerarquía con un único nivel.

Una dimensión puede tener más de una jerarquía. Un ejemplo típico, es una dimensión que representa tiempo (Fig. 5). En este caso, los niveles *bimestre* y *trimestre* no están relacionados entre sí, a pesar de que ambos están relacionados con *mes* y *año*.

Si existe más de una jerarquía, la relación que une datos de un nivel con datos de otro nivel superior debe ser confluente. Esto significa que todos los caminos que parten de un elemento *e* del nivel *E*, llegan al mismo elemento *d* del nivel *D* superior a *E*, independientemente de la jerarquía recorrida. (Fig. 5)

Si no se cumpliera la confluencia, entonces el primer trimestre que figura relacionado con 1998, podría aparecer relacionado, por ejemplo, con 1999. en este caso, al recorrer la jerarquía por trimestre, el mes de marzo de 1998 aparecería en 1999.

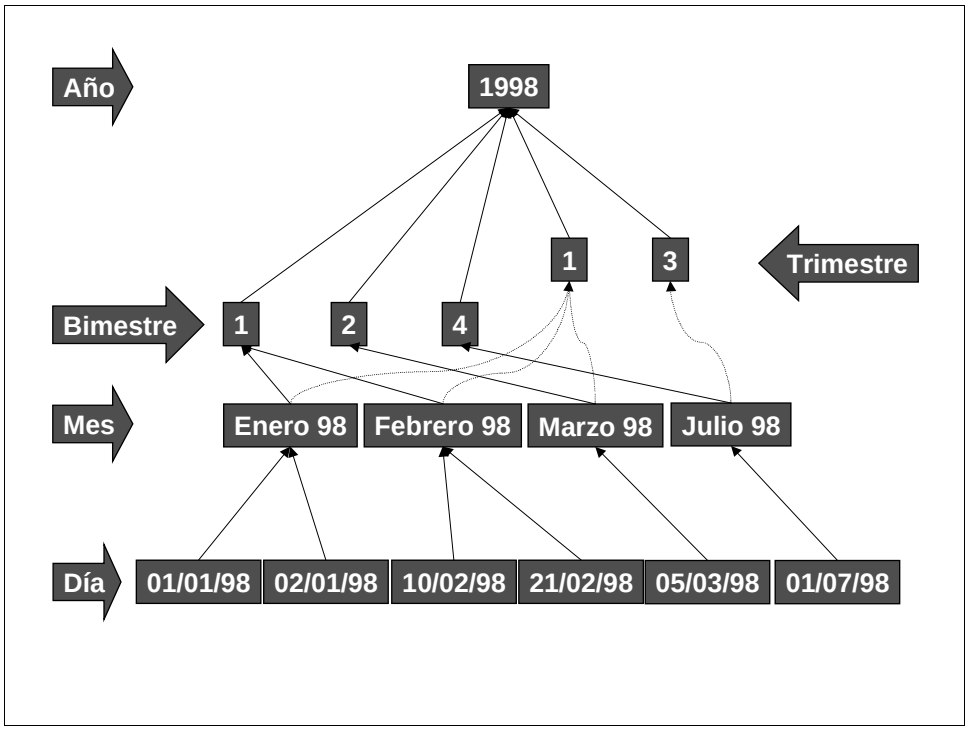

Fig. 5. Instancia de Dimensión Temporal con Jerarquías Alternativas.

2.1.3. Operaciones Multidimensionales y Espacios de Cubos.

Las operaciones multidimensionales se pueden agrupar en tres conjuntos básicos:

- De ***selección y visualización*** o ***Slice & Dice***.
- De ***Agregación***.
- De ***Relacionamiento***.

En la bibliografía revisada, no se ha encontrado consenso en la definición de ***Slice & Dice***. Sin embargo, se acepta en general, que hay tres operaciones asociadas a ***Slice & Dice***: Una que selecciona "dimensiones de trabajo" de un cubo mayor, otra que selecciona "secciones" del cubo en función de valores de las dimensiones y otra que permite "presentar" diferentes planos de un cubo. A la primera se le llama *Slice*, a la segunda *Dice* o *Filtrado* y a la tercera *Rotación*.

El grupo de las operaciones de ***Agregación*** está constituido, por operaciones que surgen de realizar "movimientos" en las jerarquías de las dimensiones. Cuando se "sube" de nivel por una jerarquía, se agrupan todos los valores del nivel original que están relacionados con el mismo valor del nivel superior, mientras que al "bajar" por la jerarquía se produce la desagregación de dichos valores. La primera operación se conoce como *DrillUp* y la segunda, su inversa, como *DrillDown*.

Cuando se realiza un *DrillUp*, se debe calcular una nueva medida en función del conjunto de los valores de las medidas que se agrupan. A esta operación se le llama *Roll-up* o *Consolidación*. Esta última operación también pertenece al grupo de operaciones de ***Agregación*** y su aplicación se traduce, típicamente, en funciones de agregación como las presentes en SQL (sum, avg, etc.).

A partir de un cubo, mediante las operaciones de ***Relacionamiento***, se puede acceder a otros datos. Si éstos últimos están en un cubo, la operación se suele llamar de *Drill-Across*, mientras que si están en el Data Warehouse o en la base operacional, la operación se suele llamar *Drill-Through*.

Dado un cubo, al aplicar operaciones de *DrillUp* o *DrillDown*, se recorre un "espacio de cubos". Dicho espacio está determinado por las dimensiones que participan en el cubo origen y la forma en que se deben realizar los cálculos con las medidas (RollUp) en cada *DrillUp*. (Fig. 6)

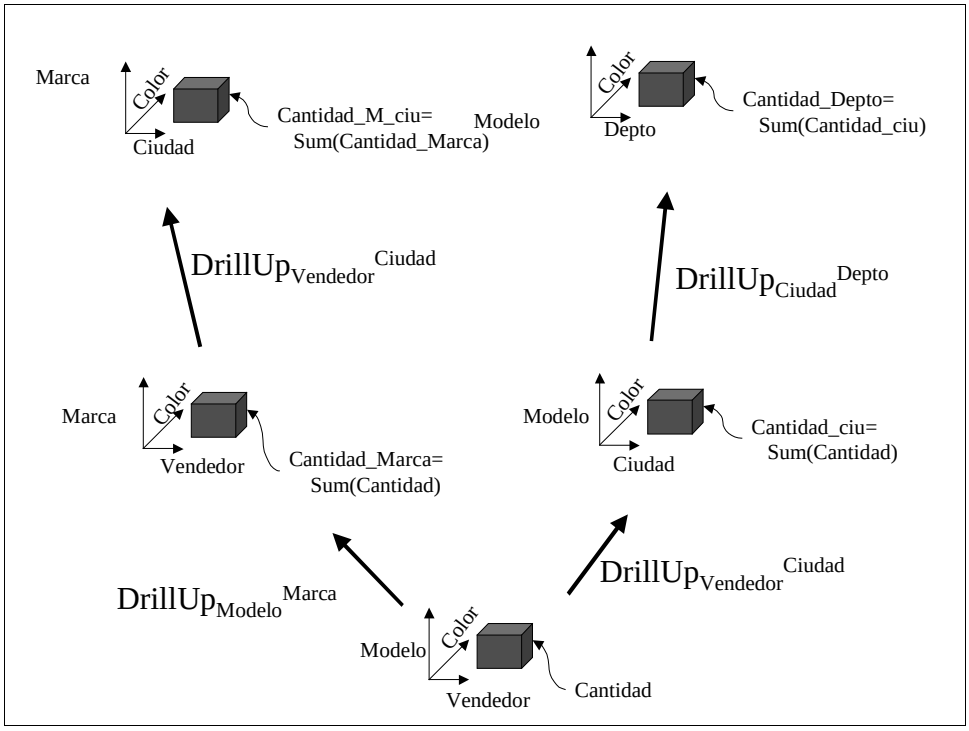

Fig. 6. Espacio de cubos generado a partir de un cubo determinado aplicando operaciones de Drill-up.

2.2. Trabajos en Diseño Conceptual.

En esta sección se hace una presentación muy breve de los modelos multidimensionales más importantes que se han encontrado en la bibliografía.

En el anexo I se describen dichos modelos con más detalle.

2.2.1. Introducción.

Los modelos multidimensionales propuestos pueden clasificarse en dos grandes familias: modelos orientados a consultas y modelos lógicos y conceptuales.

Como cualquier clasificación, la misma es arbitraria.

Los modelos orientados a consultas están más enfocados a la descripción de operaciones de manipulación de datos que a las estructuras. Sin embargo, deben basarse en alguna estructura para poder describir dichas operaciones.

Por otro lado, los modelos lógicos y conceptuales, hacen mayor énfasis en las estructuras. Los lógicos aún se enfocan a las manipulaciones, mientras que los conceptuales, prácticamente prescinden de ellas.

En la primera sección se presentan los modelos orientados a consultas, en la siguiente los lógicos y conceptuales. Por último se presentan algunas conclusiones y comparaciones entre los mismos. Las descripciones y comparaciones de los modelos, se pueden encontrar con más detalle en el anexo I.

2.2.2. Modelos Orientados a Consultas.

En esta categoría se pueden encontrar, básicamente, tres trabajos: Li y Wang ([Li96]), Gyssens y Lakshmanan [Gys97] y Agrawal, Gupta y Sarawagi ([Agr97]).

Los dos primeros, son dependientes del Modelo Relacional, o sea, se basan en el modelo relacional, por lo que su implementación en una base relacional es directa. Sin embargo, no es clara su implementación en un sistema multidimensional.

El último, si bien es posible mapearlo sobre una base relacional, presenta directamente estructuras abstractas multidimensionales.

2.2.2.1. Li y Wang.

Para Li y Wang, un esquema de cubo es un conjunto de parejas formadas por un nombre y un conjunto de atributos. Estas parejas constituyen las dimensiones del cubo. La instancia de un cubo, está dada por una pareja formada por un conjunto de relaciones, una para cada dimensión, y una función. Esta función, mapea coordenadas (tuplas) formadas por las tuplas de cada dimensión en un determinado conjunto de valores escalares.

Resumiendo la estructura, un cubo en este modelo es un conjunto de relaciones, una para cada dimensión, y una función que mapea coordenadas formadas por tuplas de esas relaciones en una única medida.

La noción de relación que utiliza es la misma que en el Modelo Relacional. El lenguaje de manipulación que propone es una extensión del Álgebra Relacional con operadores de ordenamiento y agrupamiento.

La especificación resultante es bastante intrincada, sobre todo para obtener algo muy similar a SQL desde el punto de vista del lenguaje y al Modelo Estrella ([Kim96]) desde el punto de vista de la estructura. Desde el punto de vista multidimensional, el modelo no soporta dimensionalidad genérica (sección 3.1.2). Un aspecto diferente del modelo, es que en el esquema no hay ninguna referencia la medida. Sólo se puede saber alguna información de la misma a partir de la instancia del cubo.

2.2.2.2. Gyssens y Lakshmanan.

El modelo que se propone en [Gys97] es prácticamente el mismo que el anterior desde el punto de vista estructural. Sin embargo, presenta algunas diferencias importantes en su especificación y propiedades.

La estructura básica en este modelo se llama *tabla*, aunque en realidad es un cubo. El esquema de una tabla, se define como una terna <*D, R, Pair*> donde *D* es un conjunto $\{d_1,...,d_n\}$ de nombres de dimensión, *R* es un conjunto de atributos y *Pair* es una función que mapea nombres de dimensión en subconjuntos de *R*. Esta función garantiza que los conjuntos de atributos que asocia a dimensiones diferentes, son disjuntos dos a dos.

El conjunto *M* de las medidas, está conformado por los atributos de *R* que no fueron asignados a ninguna dimensión por la función *Pair*.

La instancia de esta estructura, está dada por n+1 relaciones: una para cada dimensión y una para los atributos de *M*. Las relaciones asociadas a cada dimensión tienen un atributo extra T_{id} que sirve como identificador de tupla. La relación asociada a las medidas tiene, además de los atributos de *M* , cada uno de los atributos T_{id} de las dimensiones. Además, se cumple que los valores de los atributos T_{id} que están en la relación asociada a *M*, están en el producto cartesiano de los valores de los atributos T_{id} de cada una de las dimensiones. De esta forma, se garantiza el join sin pérdida entre las relaciones de las dimensiones y la relación de las medidas.

Resumiendo, es otra implementación del Modelo Estrella. Sin embargo, su especificación es más clara que la de Li y Wang. El lenguaje de consulta que proponen es una extensión del Álgebra Relacional con operadores de agrupamiento y funciones agregadas. Además, provee funciones para intercambiar atributos de dimensiones y medidas, con lo que provee el soporte básico para la dimensionalidad genérica.

2.2.2.3. Agrawal, Gupta y Sarawagi.

Este modelo es realmente multidimensional, o sea, sus estructuras no se describen directamente sobre las estructuras del Modelo Relacional. Sin embargo, las estructuras del modelo si se pueden mapear sobre las del Modelo Relacional.

La estructura básica es un *hipercubo* compuesto por dos elementos: un conjunto de dimensiones y una función que mapea coordenadas formadas por valores de cada una de las dimensiones en tuplas o booleanos. Una dimensión es un nombre con un dominio asociado.

El hecho de que la función mapee coordenadas en tuplas o booleanos se debe al mecanismo utilizado para implementar la dimensionalidad genérica. Si la función devuelve tuplas, entonces son todas de la misma estructura y son las medidas. En este caso, el cubo presenta la información de que para cada coordenada, las medidas que le corresponden son las que están en la tupla resultado. En cambio si devuelve booleanos, la información que presenta el cubo es que las coordenadas están relacionadas. A partir de este cubo booleano y una operación sobre los cubos, se puede obtener un cubo con información equivalente en donde cualquier dimensión puede tomar el rol de medida.

El lenguaje de consulta que se propone es un álgebra pero no es una extensión del Álgebra Relacional. Sus operaciones están orientadas a transformar dimensiones en medidas y viceversa, a seleccionar secciones del cubo y a relacionar cubos.

Como ventajas, se puede mencionar que las operaciones que sugieren están más cerca de las operaciones multidimensionales que las propuestas por los modelos anteriores. Además, la idea de "cubo booleano" que introduce, es la que toman prácticamente todos los modelos restantes para implementar la dimensionalidad genérica.

La principal desventaja, se plantea desde el punto de vista del modelado de datos, ya que no diferencia entre esquema e instancia de las estructuras.

2.2.3. Modelos Lógicos y Conceptuales.

Las propuestas que se presentan en esta sección, se caracterizan por su nivel de abstracción. Ninguna de ellas es dependiente del relacional y pueden ser presentadas por sus autores en forma totalmente independiente de la implementación.

Se presentan cinco trabajos que abarcan varios artículos cada uno: Cabbibo y Torlone ([Cab97], [Cab98], [Cab99]), Lehner ([Leh98a], [Leh98]), Golfarelli ([Gol98a], [Gol98], [Gol99]), System 42 ([Sap99a]) y DWQ ([Fra99], [Fra99a], [Cal98], [Hor99], [Hor99a], [Agr97]).

La presentación que se realiza aquí es muy somera, por más detalles se recomienda ver el anexo I.

2.2.3.1. Cabbibo y Torlone.

El modelo presenta dos estructuras básicas: Dimensiones y F-Tablas.

El esquema de una dimensión es una tupla <L,≤,R-UP> donde:

- **L** es un conjunto finito de nombres de nivel. Cada nombre de nivel tiene asociado un dominio y estos dominios de nivel son disjuntos dos a dos.
- ≤ es un orden parcial entre niveles de **L** que representa la jerarquía de la dimensión.
- **R-UP** es una familia de funciones que indica como se pueden realizar los Roll-up en la jerarquía de la dimensión.

Un esquema de f-tabla está formado por un nombre, un conjunto de atributos con un nivel asociado a cada uno y un nivel más definido como medida. Un esquema tiene la siguiente representación: $\mathbf{f\ [A_1:l_1,...,A_n:l_n]:l_0}$. En esta representación, **f** es el nombre de la f-tabla, cada $\mathbf{A_i}$ es un atributo y cada $\mathbf{l_i}$ es un nivel. El nivel $\mathbf{l_0}$ es la medida de la f-tabla.

Para definir las instancias, se define primero la noción de coordenada simbólica. Una coordenada simbólica sobre una f-tabla es una función que mapea cada nombre de atributo en un valor del dominio del nivel asociado al atributo. De esta forma, una coordenada es una tupla.

Una instancia de f-tabla, es una función que mapea coordenadas simbólicas en valores del dominio de la medida.

Un esquema multidimensional es una pareja formada por un conjunto de dimensiones y un conjunto de f-tablas sobre esas dimensiones, y una instancia de un esquema multidimensional es una función que mapea cada esquema de f-tabla en una instancia de f-tabla.

El lenguaje que presenta el modelo es un cálculo de conjuntos.

Resumiendo, las estructuras que presenta el modelo son multidimensionales y están bien definidas.

Los caminos posibles para realizar roll-up, están explícitamente en las estructuras de las dimensiones. Esto permitiría al analista mejorar la precisión de sus especificaciones.

El cálculo que se presenta, permite manipular la dimensionalidad genérica de una forma muy natural, a través del uso de f-tablas con recorrido booleano, similares a los cubos booleanos propuestos por Agrawal (2.2.2.3).

Sin embargo, el modelo adolece de un problema:

La mayoría de las veces las funciones de roll-up son extensionales, es decir, están dadas por una tabla que indica como se relaciona cada elemento de un nivel con cada elemento del nivel superior. Esto hace pensar que las funciones de roll-up deberían estar en la instancia y no en el esquema de las dimensiones.

Además, no presenta un mecanismo de especificación de restricciones de integridad.

2.2.3.2. Golfarelli, Rizzi y Maio.

En [Gol98a] el modelo es presentado como una notación para representar estructuras multidimensionales que se obtienen de la especificación Entidad-Relación de la base operativa. La estructura básica es el *esquema de hechos*.

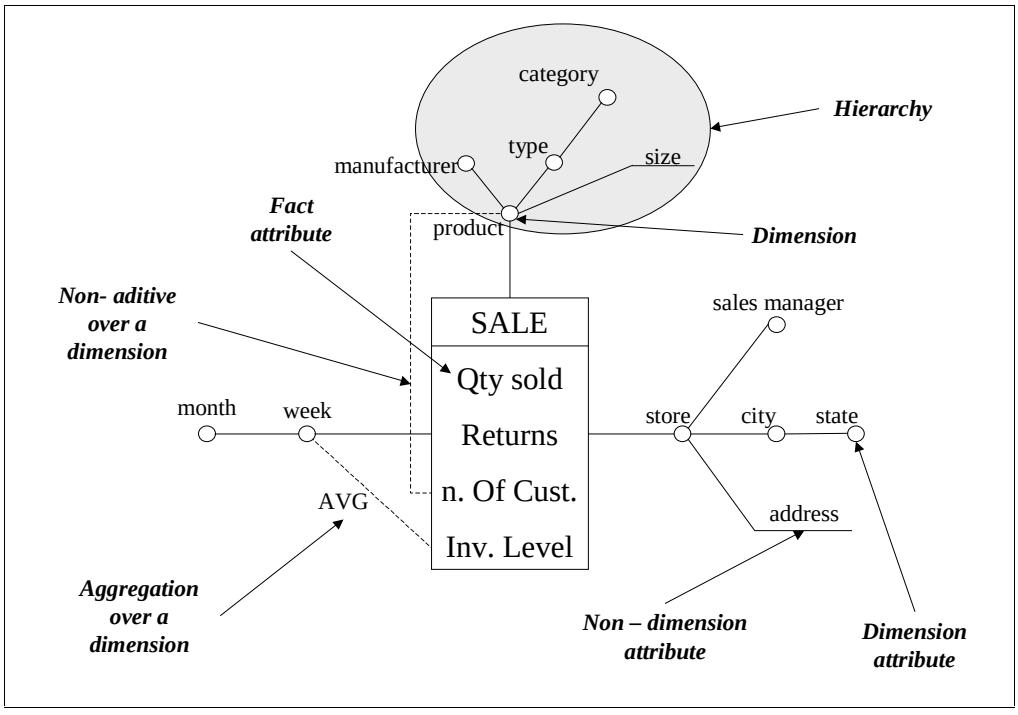

Fig. 7. Un Esquema de Hechos.

La notación tiene características interesantes:

- Permite la especificación de cuáles son las funciones de agregación mediante las que se pueden agregar las medidas. Son pocos los modelos que permiten esto. (sección 3.1.2).
- Permite visualizar gráficamente las jerarquías de las dimensiones.

Sin embargo, el modelo no soporta dimensionalidad genérica y en esa versión, no presentaba ninguna formalización del mismo. Además, no permite visualizar los posibles caminos para la realización de drill-across.

En [Gol98] y en [Gol99] , se presenta el mismo modelo pero en un marco metodológico más completo. En estos trabajos se presenta una formalización del modelo. Sin embargo, la formalización se basa en la representación gráfica, es decir, parece ser una formalización del grafo que representa el modelo más que de las estructuras multidimensionales.

En los aspectos metodológicos que se discuten, se destaca la noción de carga de trabajo y una solución para la visualización de los caminos de drill-across que tiene el aspecto de una posible metodología de integración de esquemas.

2.2.3.3. System 42.

En [Sap99a] se presenta un modelo conceptual muy particular. Es una extensión directa del Modelo Entidad-Relación, aplicada a las nociones multidimensionales.

Este modelo se construye especializando la noción de entidad y relación. Un nivel, es un tipo de entidad. El roll-up es un tipo de relación binaria y el cubo (Fact Name) es una relación N-aria, en donde las medidas se presentan como atributos de la relación.

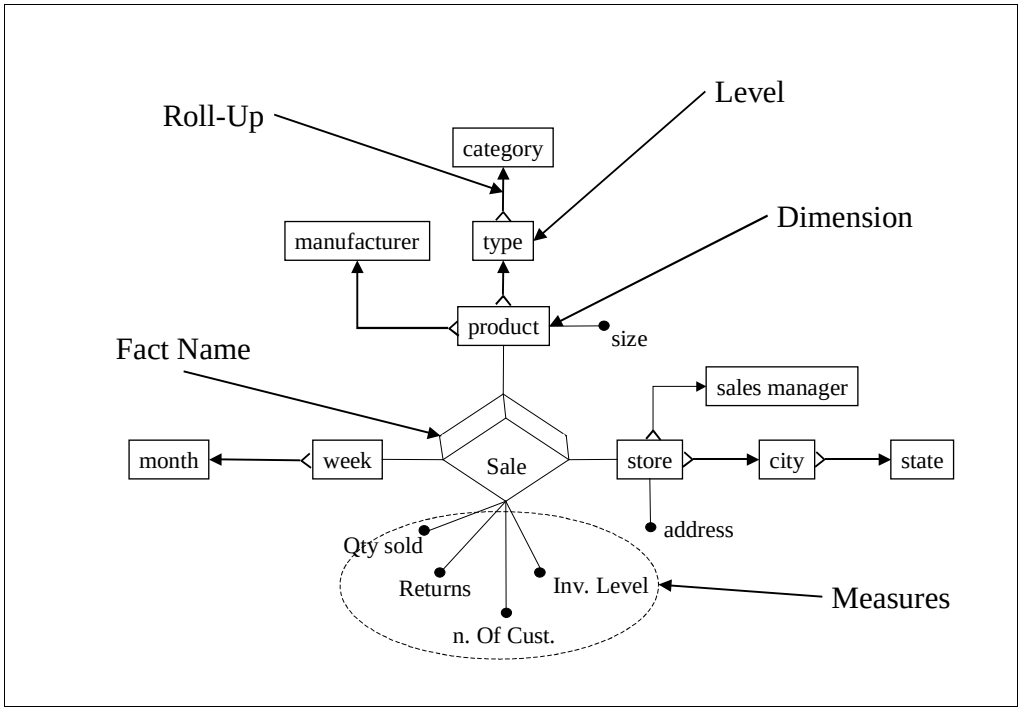

Fig. 8. Un Ejemplo en el Modelo de System 42.

El modelo presenta como principal ventaja el hecho de estar basado en otro modelo ampliamente difundido como el Entidad-Relación. Sin embargo, por este mismo motivo, no presenta un mecanismo de refinamiento o representación modular.

El modelo de Cabibbo y Torlone puede considerarse como la semántica de este modelo, interpretando los nombres de hechos como f-tablas y los niveles como niveles del modelo de Cabibbo.

2.2.3.4. Lehner.

En [Leh98a] se presenta el "Modelo Multidimensional Anidado".

Las estructuras básicas del modelo son los *Objetos Multidimensionales (MO)* que son en realidad, cubos.

Para definir estos cubos, se definen las dimensiones clasificando sus atributos en:

- **Primarios**. Hay un valor para estos atributos en cada hoja de la jerarquía de la dimensión. Permiten la identificación de los objetos de la misma y representan el nivel más desagregado de la jerarquía.
- **De Clasificación**. Para estos atributos, hay un valor en cada nodo interno o raíz de la jerarquía de la dimensión. Permiten la construcción de la jerarquía. Representan los niveles de la dimensión.
- **Descriptivos**. Describen características de los elementos de la dimensión. Pueden estar asociados a varios atributos de clasificación, pero no tienen por qué estar asociados a todos. Esto permite la manipulación de valores según un contexto definido por los atributos de clasificación. En general, representan medidas.

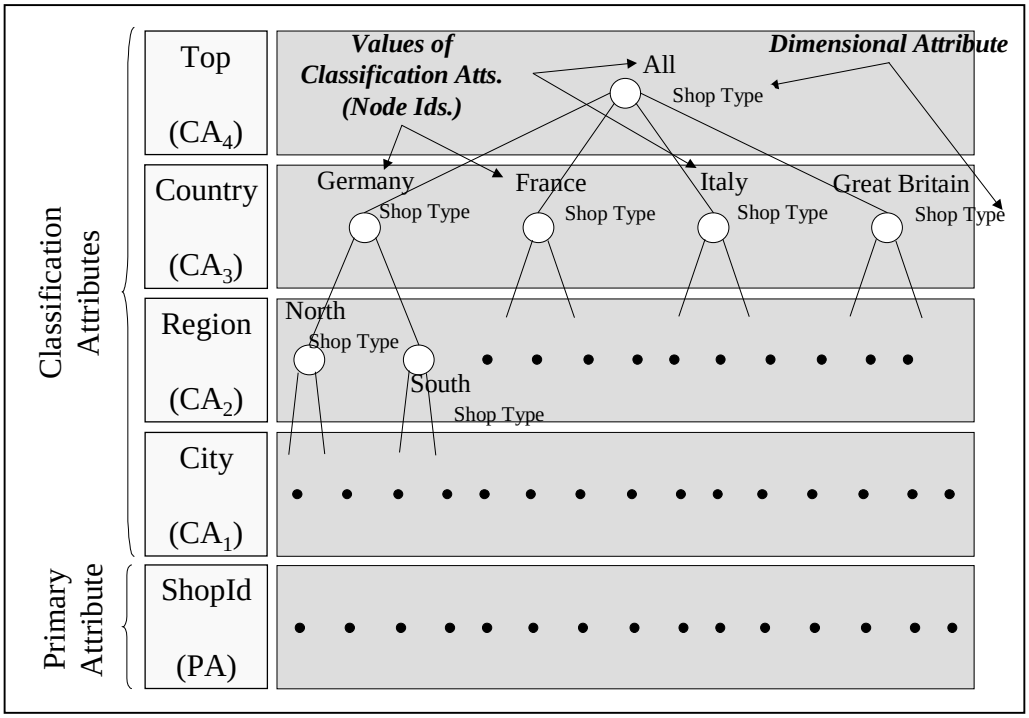

Fig. 9. Una Jerarquía de Clasificación para una Dimensión Comercios (Shops).

La jerarquía de clasificación (Fig. 9) tiene en cada nodo interno y en la raíz un valor para un atributo de clasificación y en las hojas, un valor para un atributo primario.

Un cubo (MO), se ve como un conjunto de **Objetos Multidimensionales Primarios (PMO)** que se pueden ver como cada celda de la matriz. Este conjunto de objetos se define mediante condiciones sobre los atributos de clasificación de cada dimensión que participa en el cubo. Con esto se define un PMO para cada cruzamiento de valores de las dimensiones.

A cada PMO, se le asocia un **Objeto Multidimensional Secundario (SMO)**. Cada SMO consta de un conjunto de atributos descriptivos, a partir de los cuales se pueden construir las medidas.

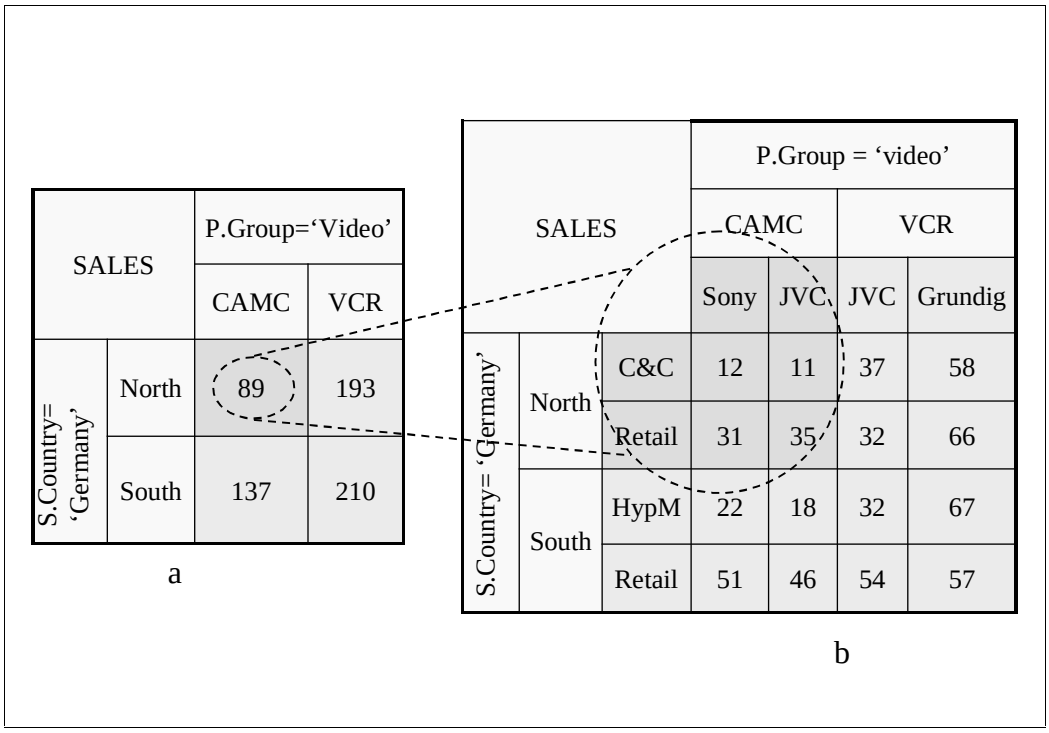

Fig. 10. Dos MO's presentados en forma tabular.

La Fig. 10 presenta dos cubos (MO´s) en forma tabular. Las celdas más claras representan el conjunto de PMO's definidos en función de las condiciones que aparecen en la primera fila y en la primera columna. En la segunda fila y columna, aparecen los valores de otros niveles relacionados con los nodos de la jerarquía que cumplen las condiciones[1]. Las celdas más oscuras, representan cada PMO del conjunto, y cada una contiene un SMO. La tabla **a** es el resultado de aplicar un Drill-up sobre dos dimensiones a la tabla **b**.

El modelo es bastante complejo. Sin embargo, presenta características muy particulares:

Permite la manipulación de medidas contextuales, es decir, que algunos elementos de la dimensión las tienen y otros no.

[1] Se pueden comparar las columnas de Fig. 10 con la jerarquía presentada en Fig. 9.

En la definición del conjunto de PMO's, se puede especificar el tipo de resumen que se puede realizar con los SMO. Si el tipo es aditivo, se puede aplicar las operaciones SUM, AVG, MIN, MAX o COUNT, mientras que si es promedio, se pueden aplicar las operación AVG, MIN y MAX y si es constante no se puede aplicar ninguna operación.

2.2.3.5. DWQ.

El modelo DWCDM presenta dos lenguajes: uno gráfico y otro formal basado en Lógicas Descriptivas (DL).

El modelo gráfico, está basado en el Modelo Entidad-Relación pero, en una forma diferente al modelo presentado en la sección 2.2.3.3. En este modelo, el mecanismo de representación de la realidad es drásticamente diferente al que normalmente se utiliza al construir un esquema Entidad-Relación[2].

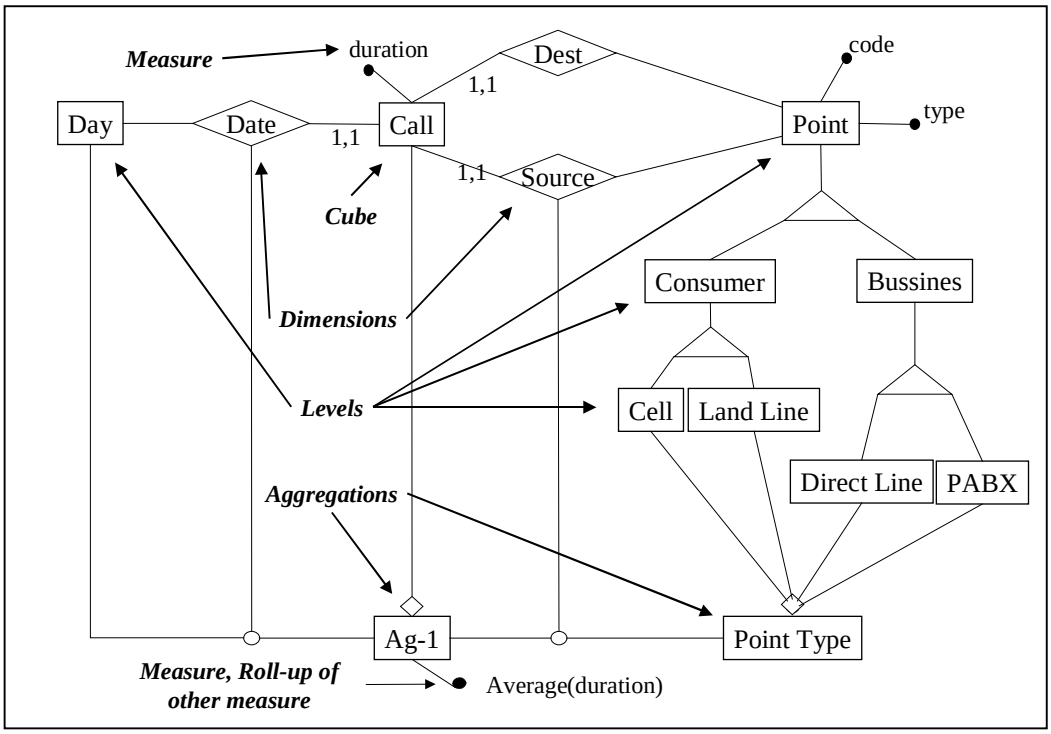

Fig. 11. Un ejemplo en DWCDM.

El lenguaje gráfico se enfoca a la descripción de agregaciones (Fig. 11) ya sea sobre cubos (como Ag-1) o sobre elementos de las dimensiones (como Point Type).

Esta representación gráfica, luego es traducida a un lenguaje basado en Lógica Descriptivas. Las Lógica Descriptivas manejan dos elementos básicos: conceptos y roles. Un concepto representa un conjunto de valores y un rol una relación binaria. Los operadores básicos son similares a los de un álgebra de conjuntos y presenta formas restringidas de cuantificación.

El punto más interesante de este modelo es que la satisfactibilidad de la especificación en DL que se obtiene al traducir el diagrama es decidible.

[2] Las diferencias en los enfoques se pueden ver comparando las figuras 7 y 8 del anexo I.

Otro detalle a resaltar es que las instancias de los esquemas se basan en el modelo de Cabibbo y Torlone (sección 2.2.3.1).

Sin embargo, no está claro en cuanto se limita la expresividad del lenguaje para conseguir esa decidibilidad.

El lenguaje formal, permite expresar algunas restricciones de cardinalidad y posiblemente, permita otras. Sin embargo, no está claro que permita la expresión de condiciones generales como que determinado valor tiene que ser menor que otro. Esto se debe a que el lenguaje que se utiliza es equivalente a una lógica modal proposicional.

Otro inconveniente es que no se ha encontrado en la bibliografía consultada, una descripción suficientemente detallada del lenguaje gráfico.

2.3. Conclusiones.

El trabajo inicial de Cabibbo y Torlone es fundamental en el desarrollo de los modelos posteriores. Realmente captura las aspectos multidimensionales más básicos.

Todos los modelos que se analizaron presentan aspectos interesantes. Sin embargo:

- De los modelos realmente conceptuales (Golfarelli, System 42, DWQ y Lehner) ninguno soporta la dimensionalidad genérica.
- Tampoco soportan restricciones explícitas. Si bien la base formal del modelo propuesto por el grupo DWQ permitiría la expresión de restricciones, las que se manejan en dicha propuesta son muy limitadas y no es claro que se puedan expresar otras restricciones más generales.

En las siguientes secciones, se presenta CMDM, un modelo conceptual que soporta dimensionalidad genérica y que provee un lenguaje de restricciones de integridad.

Capítulo 3. CMDM: Estructuras Básicas.

En la primer sección de este capítulo se presentan los fundamentos del modelo, describiendo las diferentes vertientes consideradas a la hora de diseñar un modelo de datos conceptual para bases multidimensionales.

Luego se presentan las estructuras a través de un caso de estudio.

En la tercer sección se presenta el lenguaje gráfico en detalle y en la cuarta sección una formalización de las estructuras necesarias.

En la última sección se discuten algunos ejemplos sobre el caso de estudio que permiten ver las limitaciones de la propuesta y exponer la necesidad de una extensión de la misma.

3.1. Fundamentos del Modelo.

A la hora de diseñar un modelo de datos conceptual para un tipo determinado de sistema de información, es necesario considerar un conjunto de elementos relativos a tres aspectos: elementos relativos al modelado conceptual, al tipo específico de sistema de información y al proceso de diseño de ese tipo de sistema de información.

A continuación se presentan estos aspectos para el caso particular de un sistema de información basado en un Data Warehouse y en aplicaciones OLAP.

3.1.1. Consideraciones acerca de los Modelos Conceptuales.

3.1.1.1. Principios Básicos.

En [LZ92] se presentan dos principios en los que se debe basar un esquema conceptual:

1. **Principio del 100%.**

   Todas las leyes y reglas que gobiernan el Universo de Discurso deben estar definidas en el esquema conceptual. Ninguna de estas leyes o reglas debe convertirse en parte de una aplicación o ser distribuida entre diferentes programas de aplicación. La formulación de estas reglas y la recuperación y manipulación de la información deben estar estrictamente separadas.

2. **Principio de Conceptualización.**

   El esquema conceptual debe referirse exclusivamente a reglas del universo de discurso. Las reglas que gobiernan la implementación del sistema de información no deben convertirse en parte del esquema conceptual.

En base a los principios anteriores, también en [LZ92] se recopilan algunos requerimientos que se exigen a un modelo conceptual. Los mismos se presentan en la siguiente sección.

3.1.1.2. Requerimientos para los Modelos Conceptuales.

No es obligatorio que un modelo conceptual cumpla totalmente, todos los requerimientos presentados en esta sección. Sin embargo, el grado en que cumple cada requerimiento permite determinar el grado conceptual del modelo.

1. Independencia de la Implementación.

El esquema conceptual no debe contener ningún aspecto de la implementación, como pueden ser la representación de datos, la organización física de los datos y los mecanismos de acceso, etc.

2. Abstracción.

Sólo se deben representar aspectos dinámicos y estáticos generales de un sistema de información y de un Universo de Discurso en una especificación de requerimientos, fundamentalmente los aspectos que no están sujetos a cambios frecuentes.

3. Formalidad.

Las descripciones deberían ser establecidas mediante algún formalismo, sin ambigüedad y que pueda ser comprendido y analizado por un procesador razonable. El formalismo debe soportar una teoría semántica rica que permita relacionar las descripciones en el formalismo con los objetos del mundo real que están siendo modelados.

4. Constructibilidad.

Un esquema conceptual se debe construir de tal forma que sirva como medio de comunicación entre los analistas y los usuarios y debe ser capaz de manejar dominios muy complejos de una forma relativamente clara.

5. Facilidad de Análisis.

Un esquema conceptual debe ser analizado para determinar su ambigüedad, completitud y consistencia. Una especificación es ambigua si se le pueda asociar más de una interpretación. Completitud y consistencia requieren la existencia de estas nociones en el modelo, contra las que se pueda testear la especificación.

6. Trazabilidad.

Se refiere a la posibilidad de seguir la pista de los objetos que se especificaron a través del proceso de desarrollo. Esto es que se puedan seguir los cambios de los objetos desde la especificación hasta la implementación, pasando por el diseño.

7. Ejecutabilidad.

Esta propiedad es importante a los efectos del testeo de la validez de la especificación.

3.1.2. Consideraciones acerca de las Aplicaciones Olap.

E. F. Codd, creador del Modelo Relacional, publica en 1993 un artículo técnico realizado para la empresa Arbor Software en donde propone doce reglas que debe seguir una aplicación para ser considerada OLAP ([Cod93]). Una de las críticas que recibió este artículo, entre otras (ver [Tho97] y [Kim96]), es que dentro de las reglas se mezclan elementos conceptuales, lógicos y de implementación. Aún así, hay dos reglas que describen propiedades deseables en la aplicaciones OLAP y que tienen carácter conceptual:

Visión Conceptual Multidimensional.

La visión de los datos de la empresa que tiene el usuario es multidimensional, por lo que el analista, debe ser capaz de trabajar con el modelo con la misma visión.

Dimensionalidad Genérica o Generalidad Dimensional.

Se debe poder realizar cualquier operación relativa a los aspectos dimensionales sobre cualquier dimensión. Esto implica que cualquier dimensión es susceptible de ser considerada una medida y viceversa.

Resumibilidad.

Por otro lado, en [Kim96] se discute sobre la aditividad (aditivity) de las medidas. Este mismo tema es tratado con más generalidad en [Len97] con el nombre de resumibilidad (summarizability). El punto de discusión es el siguiente: ¿Cualquier medida soporta cualquier tipo de operación de agregación sobre ella generando datos coherentes desde el punto de vista de la realidad?

El siguiente ejemplo puede aclarar un poco más el problema.

Un servicio meteorológico tiene una aplicación OLAP con los datos de temperatura tomados en las diferentes estaciones meteorológicas en Uruguay a lo largo de un año. Los datos están en un cubo que tiene como dimensiones las estaciones y la fecha. Como medida, este cubo tiene la temperatura tomada en cada estación.

Dentro de los análisis posibles, se desea hacer un roll-up de *estación* a *departamento*, sumando las medidas. ¿Es razonable el resultado de esta operación?.

Para verlo más claramente, se puede considerar que en el departamento de Montevideo hay por lo menos dos estaciones: una en el Prado y otra en la zona de Melilla. Entre los datos, puede figurar que en determinado día de enero de 1999, en el Prado se tomo una máxima de 35 grados Celsius y en Melilla una máxima de 32 grados Celsius. Al realizar la operación anterior, se estaría registrando que para el departamento de Montevideo, en esa fecha hubo una máxima de 65 grados Celsius, lo cual es claramente, una temperatura ambiente absurda.

La suma, es una operación sin sentido en el contexto de esos datos. En todo caso, una operación más coherente debería ser el promedio de las temperaturas.

La resumibilidad es una característica de interés a la hora de especificar un cubo porque expresa en algún sentido, una noción de correctitud de los datos.

3.1.3. Consideraciones acerca del Proceso de Diseño.

Durante los últimos 30 años, se han propuesto diferentes metodologías de análisis y diseño de sistemas de información. Sin embargo, aún no hay ninguna metodología específica para el análisis y diseño de sistemas de Data Warehousing que sea ampliamente aceptada.

Entre otros elementos comunes a las diversas metodologías, hay uno de particular interés: la representación de información a diferentes niveles de abstracción.

Sería deseable que cualquier modelo conceptual tuviera mecanismos de soporte para esta característica.

3.1.4. CMDM y los Fundamentos.

En el diseño de CMDM se intenta aplicar en lo posible, la mayoría de las ideas anteriores.

CMDM está tiene dos componentes: Un lenguaje gráfico para el diseñador y un modelo formal detrás. Esto facilita el cumplimiento de los requerimientos de los modelos conceptuales presentados anteriormente. Si un requerimiento no se cumple en un nivel dado, debería cumplirse en el otro.

Entre los requerimientos más complejos está la ejecutabilidad de la especificación. Para esto se hace necesaria la implementación de un sistema que sea capaz de tomar esta especificación y generar un sistema realmente operativo en algún ambiente de ejecución.

El objetivo de CMDM es especificar bases multidimensionales y eso significa especificar datos para OLAP y Data Warehousing.

La dimensionalidad genérica es una propiedad más a tener en cuenta a la hora de realizar la especificación a pesar de que casi ningún sistema la maneja. Sin embargo, es la realidad la que debe indicar si esta propiedad se debe imponer o no en un modelo concreto. CMDM es capaz de soportarla en una forma relativamente natural.

CMDM propone nuevas estructuras llamadas Relaciones Dimensionales, más abstractas que los cubos tradicionales. Estas estructuras permiten la especificación de conjuntos de cubos con características comunes. De esta forma el modelo provee una estructura que permite manejar diferentes niveles de abstracción.

Con respecto a los otros modelos, CMDM puede verse como una modificación y extensión del modelo MD propuesto en [Cab97] .

3.2. Estructuras en CMDM.

El objetivo fundamental de CMDM es permitir la especificación de una determinada realidad en términos multidimensionales, soportando al menos las estructuras presentadas en la sección 1.1.

Para lograr esto, CMDM presenta tres estructuras básicas:

- Niveles.
- Dimensiones.
- Relaciones Dimensionales.

En esta sección se presentan las ideas básicas detrás de estas estructuras y su notación gráfica.

3.2.1. Niveles.

Un nivel representa un conjunto de objetos que son de un mismo tipo. Cada nivel debe tener un nombre y un tipo. Conceptualmente, no tiene diferencia con cualquier elemento del Modelo Entidad Relación que pueda ser considerado un conjunto de Entidades.

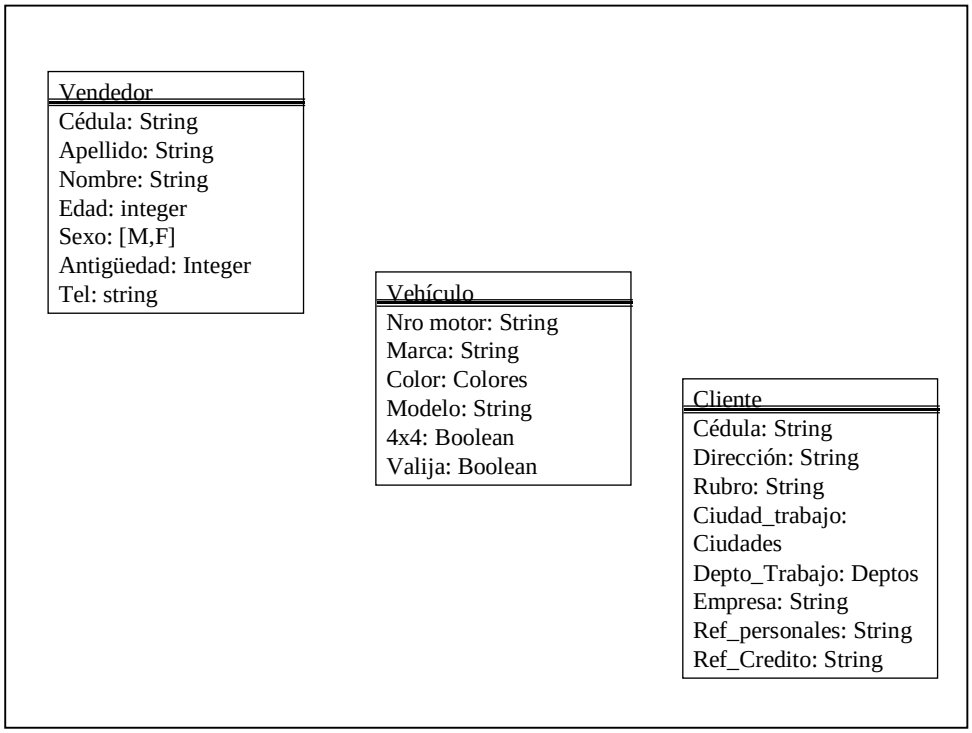

Fig. 12. Tres niveles. Un nivel debe tener un nombre y un tipo.

Para representar el esquema de un nivel se utiliza un rectángulo que contiene el nombre y la estructura (o el nombre) del tipo de ese nivel.

En la figura anterior, se pueden ver tres niveles. El primero representa un conjunto de vendedores de los que se conoce un determinado número de atributos. Análogamente, el segundo representa un conjunto de vehículos y el tercero un conjunto de clientes.

3.2.2. Dimensiones.

Una dimensión está determinada por una jerarquía de niveles. La instancia es una jerarquía de elementos de esos niveles (Fig. 4 y Fig. 5).

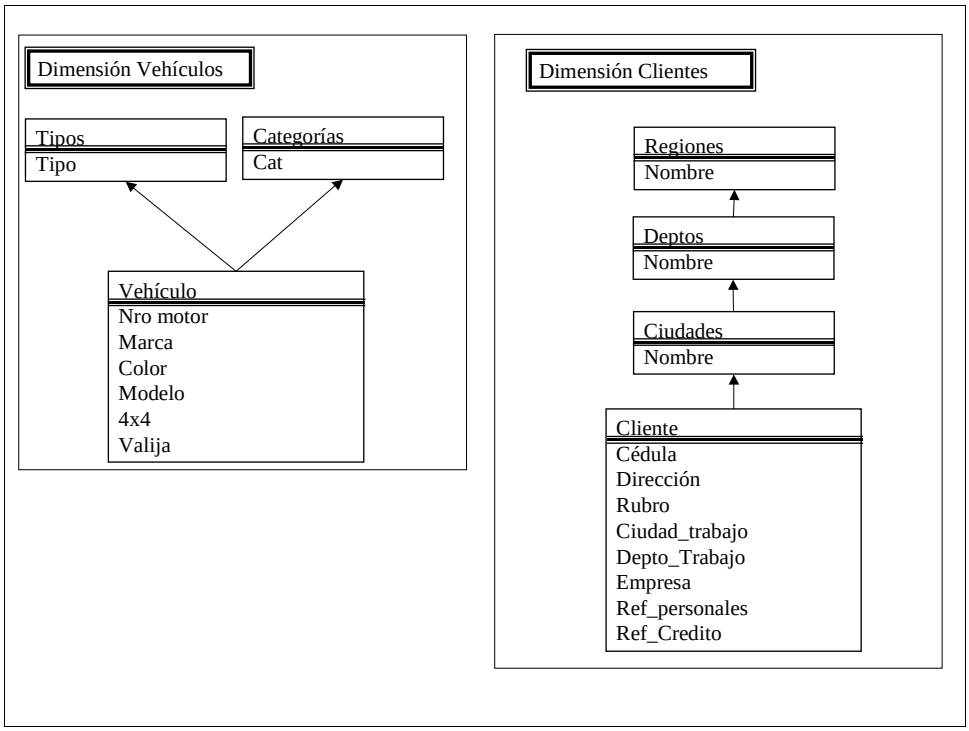

Fig. 13. Dimensiones Clientes y Vehículos.

De esta forma, el esquema de una dimensión está representado por un rectángulo dentro del cuál aparece un nombre para la dimensión y un grafo dirigido en donde los nodos son los niveles que participan de esa dimensión (Fig. 13).

3.2.3. Relaciones Dimensionales.

Una relación dimensional representa el conjunto de todos los cubos que se pueden construir a partir de los niveles de un conjunto dado de dimensiones. El dibujo de la Fig. 6 se puede interpretar como una instancia de una relación dimensional.

Se asume que en cada uno de los cubos que pertenecen a la instancia de la relación dimensional, debe aparecer al menos un nivel de cada una de las dimensiones que participan en la relación.

En CMDM, un cubo es una función que va del producto cartesiano de las instancias de los niveles en los booleanos. De esta forma, cualquier nivel puede cumplir el rol de medida.

Por lo tanto, el esquema de una relación dimensional está dado por un grafo en forma de estrella. El nodo central es de forma oval y tiene el nombre de la relación dimensional y los nodos “satélite” son rectangulares y tienen el nombre de cada una de las dimensiones que participan de la relación (Fig. 14).

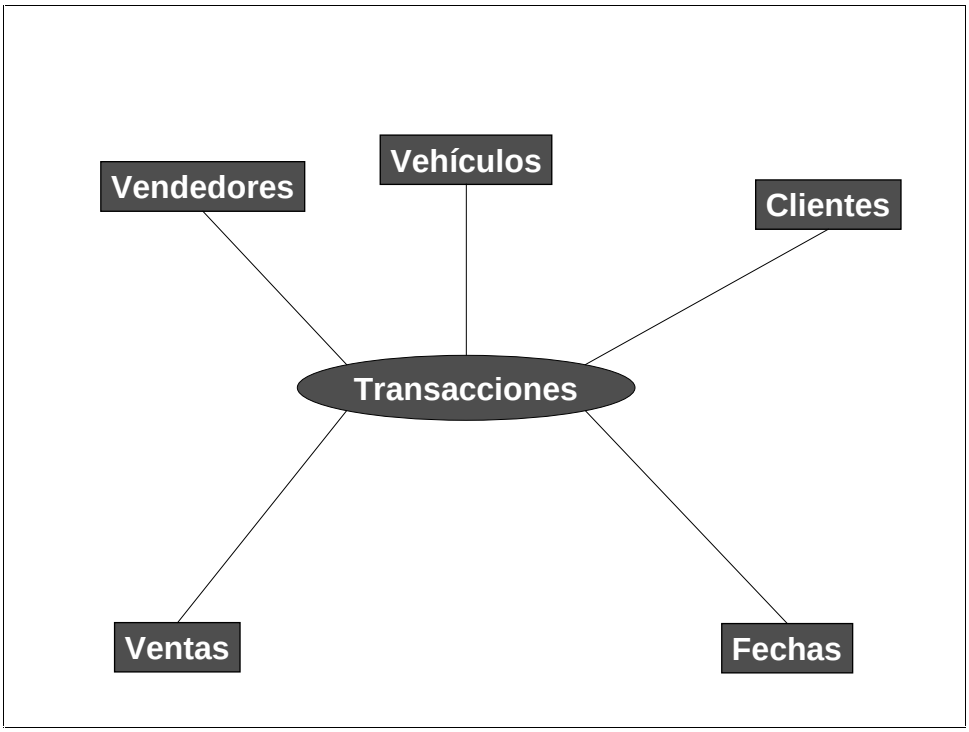

Fig. 14. Relación Dimensional para Representar las Transacciones (Ventas).

3.2.4. Conclusión.

Sobre las tres estructuras presentadas anteriormente se construyen las especificaciones en CMDM.

En la siguiente sección se presenta un caso de estudio realizado solamente con estas estructuras, fundamentalmente, para mostrar sus limitaciones.

3.3. Un Caso de Estudio en CMDM.

En esta sección se presenta un caso de estudio en que se pueden ver las estructuras básicas de CMDM. Primero se presenta una descripción de la realidad y luego, el diseño.

3.3.1. Una Empresa de Venta de Automotores.

Una empresa vende vehículos automotores en diferentes puntos del país.

De los vendedores se conoce el nombre y apellido, la cédula de identidad, el sexo, la edad, la antigüedad en la compañía y un teléfono.

Los vendedores trabajan para determinada sucursal que está en alguna ciudad de algún departamento.

Los departamentos están clasificados en 5 regiones: norte, sur, este, oeste y centro.

De los vehículos que se venden se conocen la marca, el color, el modelo y el nro. de motor. Además, la clase de vehículo está dada por cuatro propiedades particulares:

- El tipo del vehículo que puede ser *auto*, *camioneta, camión* o *tractor*.
- La categoría del vehículo según su uso puede ser *familiar* o *utilitario*.
- La tracción, puede ser 4x4 o no.
- Tipo de valija, esto es si es sedán (con valija, efectivamente) o no.

De los clientes se lleva también un cierto conjunto de datos:

Personales: nombre y apellido, cédula, dirección en la ciudad donde vive, ciudad donde vive, departamento.

Laborales: Rubro de Actividad (Un rubro válido es *pasivo*), ciudad o paraje donde trabaja, departamento donde trabaja, empresa.

Se desea analizar la información de qué vehículo fue comprado por qué cliente a qué vendedor. Entre otros datos interesa a qué precio se realizó la transacción en qué fecha y si fue a crédito o contado. También interesa tener resúmenes por sucursal, ciudad, etc., por categoría y tipo de vehículo y por ciudad donde vive el cliente (puede ser depto. donde vive o región, etc.)

Hay algunos datos adicionales de la realidad que se deben tener en cuenta:

- Los vehículos utilitarios son camionetas, camiones o tractores, pero nunca son autos.
- Los vehículos familiares son autos o camionetas pero nunca son tractores o camiones.
- Los vendedores se especializan en vehículos de determinada categoría, o sea, o bien sólo venden utilitarios o bien sólo venden vehículos familiares pero no hay vendedores que vendan las dos categorías de vehículos ya que la empresa no lo permite de ninguna manera.

3.3.2. Diseño.

En esta sección se siguen ciertas líneas para construir el diseño multidimensional de la realidad descripta en la sección anterior. Esas líneas son arbitrarias, no pretenden fijar una metodología específica de diseño.

3.3.2.1. Dimensiones.

A partir de la descripción de la realidad que se presentó anteriormente, hay ciertas dimensiones que aparecen claras, aunque podría no estar clara la estructura interna de cada dimensión.

En principio, se construye una dimensión para cada objeto que participa en el problema.

De esta forma, las dimensiones que se obtienen inicialmente son:

- *Vendedores.*
- *Vehículos.*
- *Clientes.*

Además, está claro que se desea estudiar las transacciones realizadas entre la empresa y los clientes de acuerdo a las fechas. Para ello, es claro que se necesitará una dimensión que represente esas transacciones, a la que se llamará *Ventas*.

Una duda que puede quedar es qué sucede con las fechas de las ventas. Las fechas podrían dejarse como un atributo de la venta, sin embargo, es usual que se realicen análisis de acuerdo a determinadas fechas o épocas como puede ser análisis por meses, o fechas típicas de regalos como navidad, etc. Por este motivo, es útil poner la fecha como una jerarquía de agregación. Lo usual es ponerlo como una dimensión más.

De esta forma, en el problema se han identificado cinco dimensiones:

- *Vendedores.*
- *Vehículos.*
- *Clientes.*
- *Ventas.*
- *Fechas.*

3.3.2.2. Niveles de las Dimensiones.

Los datos que conocemos sobre los vendedores son los siguientes:

- Nombre
- Apellido
- Cédula
- Sexo
- Edad
- Antigüedad
- Teléfono
- Sucursal
- Ciudad de la Sucursal
- Departamento de la Ciudad de la Sucursal.
- Región del Departamento.

Para determinar los niveles iniciales de la dimensión vendedores hay que considerar que dicha dimensión debe tener un nivel “mínimo” en donde exista un identificador para cada elemento de la dimensión. Luego, los niveles restantes se construyen de forma que la relación entre un nivel y el superior sea N:1 ya que cada nivel debe agregar varios elementos del nivel inferior.

Para determinar en qué nivel se pone un determinado atributo, debe utilizarse la información sobre los requerimientos de información. Otra alternativa es poner cada atributo en un nivel diferente. En este caso se sigue una alternativa más conservadora separando niveles sólo cuando los atributos son relevantes para los requerimientos de agregación.

De esta forma, la dimensión vendedores se puede ver en la Fig. 15.

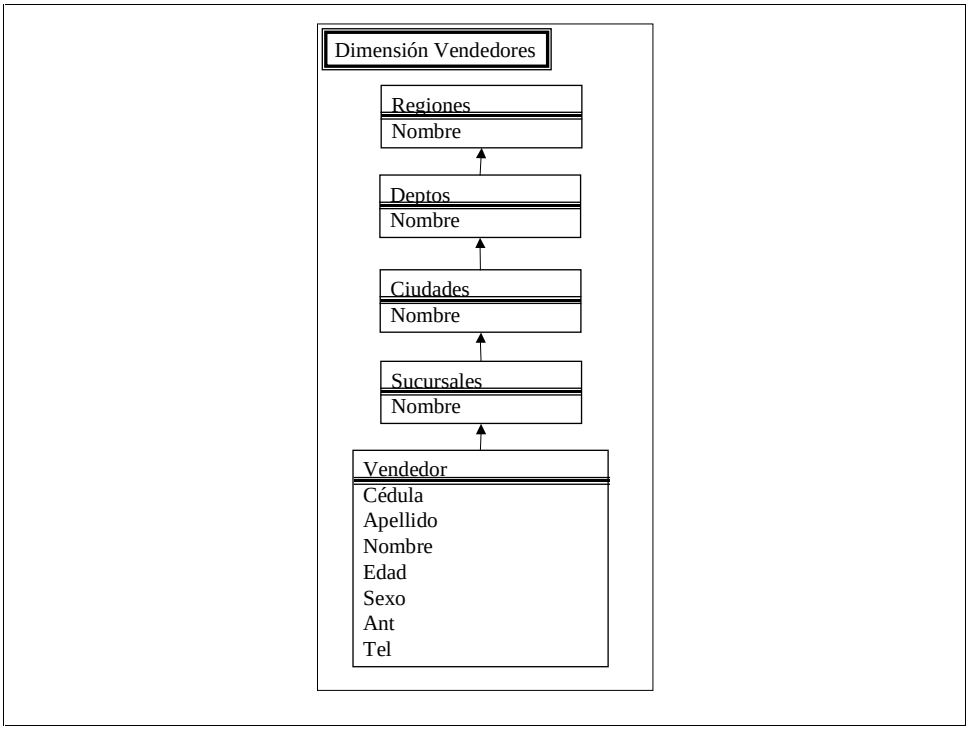

Fig. 15. Dimensión Vendedores.

Siguiendo un razonamiento similar se pueden obtener las dimensiones restantes. (Fig. 16 y Fig. 17).

Por simplicidad o claridad, en las dimensiones se omitieron los tipos de los atributos.

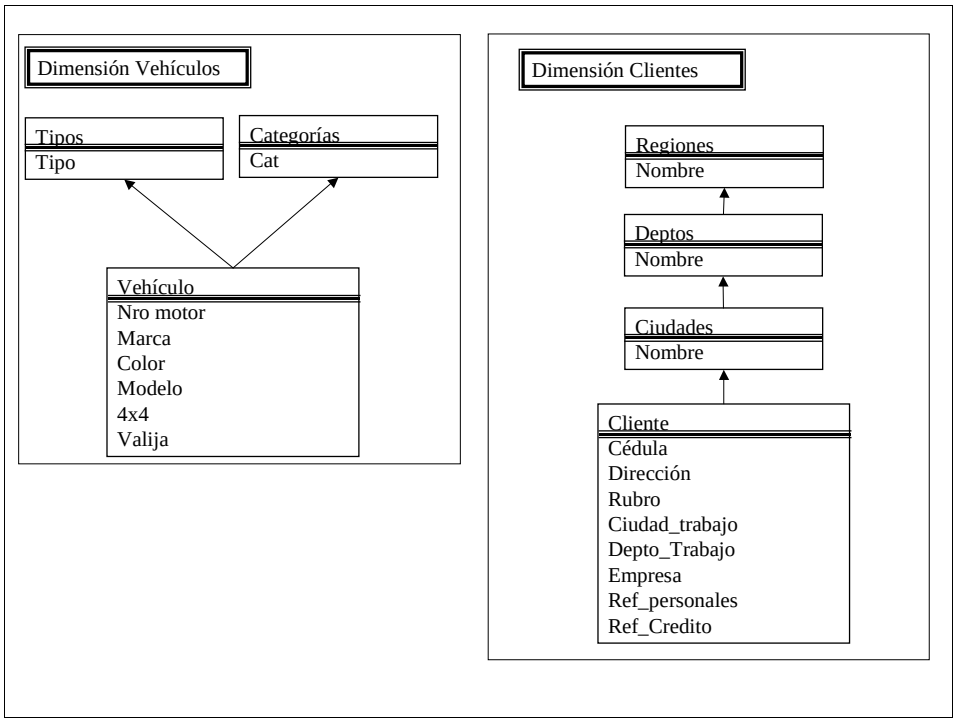

Fig. 16. Dimensiones Clientes y Vehículos.

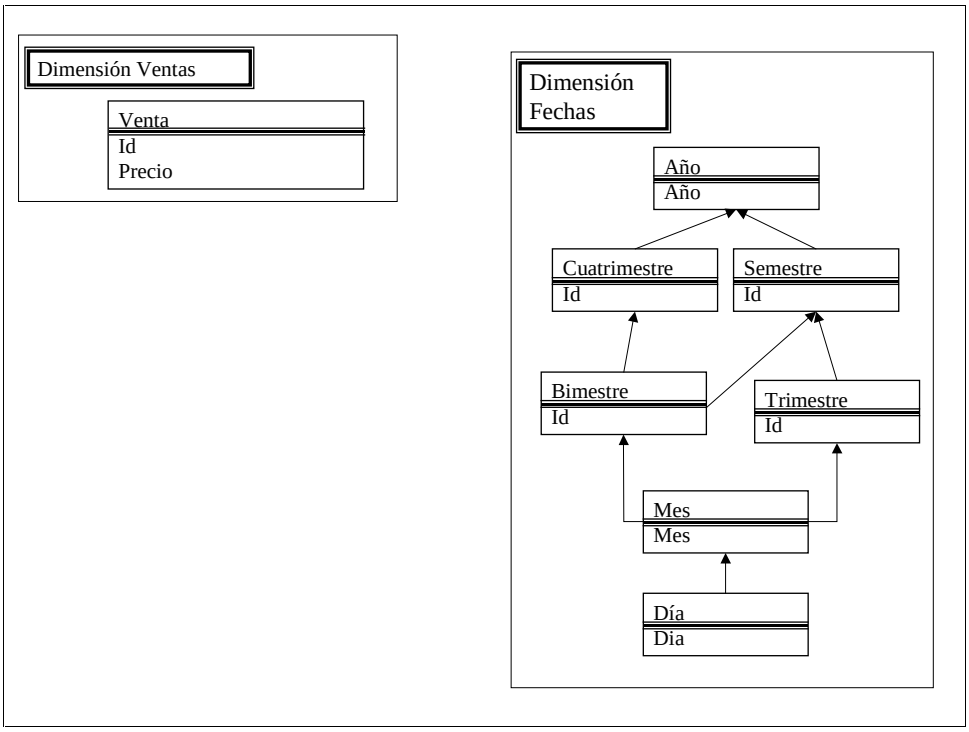

Fig. 17. Dimensiones Ventas y Fechas.

3.3.2.3. Relaciones Dimensionales.

Las relaciones dimensionales representan el conjunto de cubos que se pueden construir tomando al menos un nivel de las dimensiones participantes.

De acuerdo con la realidad planteada, los cubos de interés (al menos al momento de contacto con el usuario) tienen como dimensiones participantes a todas las dimensiones identificadas en el problema. Por esto, se puede representar todo con una única relación dimensional: la misma que fue presentada en la sección anterior (Fig. 14).

3.3.3. Conclusiones.

Si bien se han presentado todas las estructuras de CMDM, las mismas no son suficientes para expresar todos los elementos planteados en el caso de estudio.

Los diagramas no tienen en cuenta elementos como los siguientes:

- *Los vehículos utilitarios son camionetas, camiones o tractores, pero nunca son autos.*
- *Los vehículos familiares son autos o camionetas pero nunca son tractores o camiones.*
- *Los vendedores se especializan en vehículos de determinada categoría, o sea, o bien sólo venden utilitarios o bien sólo venden vehículos familiares pero no hay vendedores que vendan las dos categorías de vehículos ya que la empresa no lo permite de ninguna manera.*

Todas estas frases, expresan condiciones que deben cumplir las instancias. En particular, las dos primeras deben ser impuestas sobre los elementos de categoría y vehículos que estén relacionados por la función de DrillUp. La última es una condición que deben cumplir los cubos de la relación dimensional *Transacciones*.

Para solucionar este punto, se necesita un lenguaje de restricciones de integridad sobre las estructuras definidas anteriormente.

Por otro lado, sólo se ha realizado una presentación somera de las estructuras y no una formalización de las mismas. A la hora de estructurar una definición formal del lenguaje de restricciones, la formalización de las estructuras se vuelve imprescindible.

En la siguiente sección, se presenta una formalización de las estructuras anteriores.

3.4. Especificación del Modelo.

En la primera sección, se presenta un resumen del lenguaje de especificación para facilitar la lectura de la propia formalización del modelo. Luego se presentan los esquemas de cada una de las estructuras. Para ello además de la especificación, se incluye una pequeña fundamentación (más bien informal) de la correctitud de esa especificación. Por último se presenta la especificación de las instancias de estos esquemas y un pequeño ejemplo.

3.4.1. El Lenguaje de Especificación.

El lenguaje utilizado para especificar tanto esquemas como instancias pretende ser intuitivo para quien maneje los conceptos básicos sobre conjuntos y lenguajes formales.

El lenguaje, hace uso intensivo de conjuntos definidos por comprensión y por extensión. Se asume como predicados la pertenencia de elementos a conjuntos, la igualdad y comparación de conjuntos (inclusión amplia y estricta) y también de elementos.

Las expresiones de la forma <**a, b,...**> representan elementos de un producto cartesiano con componentes de nombre **a, b**, etc. Si **A** es una expresión para un elemento de ese producto cartesiano, **A.a** es una expresión para la componente **a** de ese elemento.

La expresión **A** $\rightarrow$ **B** representa el conjunto de las funciones que van del conjunto **A** en el conjunto **B**.

En el contexto de una expresión como la anterior, si la expresión **A** es un conjunto definido por comprensión, las variables que se usan en la definición de los elementos del conjunto podrán ser usada en la expresión **B** y serán consideradas las mismas variables. De esta forma, la siguiente expresión:

$\{x / x \in Pares\} \rightarrow \{ y / \exists(k \in Impares).y = x^k \}$

representa el conjunto de las funciones que mapean un número par en alguna de sus potencias impares. Observar que el símbolo $\rightarrow$ se usa tanto para denotar el conjunto de las funciones en un contexto de conjuntos como la implicación en un contexto de fórmulas lógicas.

Las expresiones de la forma **F(b,...)** representan uno de los dos elementos siguientes:

- O bien, la aplicación de la función representada por la expresión **F** a lo que representan las expresiones que están entre los paréntesis.
- O bien, la aplicación de una macro de nombre **F** a las expresiones que están entre paréntesis.

La expresión **Set(A)** es un caso particular de lo anterior y representa el conjunto de los subconjuntos finitos del conjunto **A**.

El símbolo $\equiv$ se utiliza para dar definiciones de macros. Lo que está a la izquierda puede ser un nombre o un nombre seguido por una lista de expresiones separada por comas y entre paréntesis. De esta forma, la expresión

$Menores(A,C) \equiv \{x / x \in A \wedge x < C \}$

es una macro que cuando se aplica dos expresiones, se expande como la expresión de la derecha sustituyendo la expresión *A* por la primer expresión y la expresión *C* por la segunda. Se asume que en la expansión siempre se sustituyen todas las ocurrencias de los parámetros. La expresión de la derecha, puede ser cualquier expresión del lenguaje, ya sea un conjunto por comprensión o una fórmula lógica. Los parámetros también pueden ser cualquier expresión que sea conveniente, incluyendo una expresión lógica ya que con las macros sólo se manipula el texto de las expresiones.

Las fórmulas lógicas que se utilizan son con cuantificación relativa, es decir, se indica sobre qué conjuntos toman valores las variables que se cuantifican.

3.4.2. Especificación de CMDM.

3.4.2.1. Esquema Multidimensional.

Una *Base de Datos Multidimensional*, o *Base Multidimensional* es un conjunto de *Cubos* construidos sobre determinadas *Dimensiones*, que a su vez están construídas sobre determinado conjunto de *Niveles*, tales que cada nivel tiene un tipo determinado.

De lo anterior, un *Esquema Multidimensional* está formado por los siguientes elementos:

- Un identificador.
- Un Conjunto de *Declaraciones* de *Niveles*.
- Un Conjunto de *Dimensiones* construidas sobre los niveles declarados.
- Un Conjunto de *Relaciones Dimensionales* construidas sobre las dimensiones del esquema.

Definición 1. Esquema Multidimensional.

E es un *Esquema Multidimensional* si pertenece al siguiente conjunto:

$$\textbf{DimSch} \equiv \{ \langle \textbf{Name, Decls, Dims, Rels} \rangle \mid \textbf{Name} \in \textbf{Strings} \wedge \textbf{Decls} \in \textbf{Declarations} \wedge \textbf{Dims} \in \textbf{Set(Dimensions(Decls))} \wedge \textbf{Rels} \in \textbf{Set(Relations(Dims))} \}$$

En las siguientes secciones se verá qué es cada elemento de un esquema.

3.4.3. Declaraciones (Decls).

Una declaración es una forma de asociar un tipo, entendiendo por tipo la noción de conjunto de valores con operaciones asociadas (o métodos). Esto se logra siguiendo la misma línea de [Cab97] : *Types* es una familia de tipos tal que a un string del conjunto *Level_Names* determinado se le asocia un tipo determinado.

La instancia de cada nivel, es un conjunto de elementos del tipo asociado.

En la especificación que se presenta, ETYPE es el conjunto de las expresiones de tipo. Dicho de otra forma, es el lenguaje de los tipos que pueden estar asociados a un nivel.

Definición 2. Declaraciones de Nivel.

Decls es un conjunto de declaraciones si pertenece al conjunto **Declarations** definido de la siguiente forma:

Declarations ≡ { <Level_Names,Type> /
Level_Names ∈ Set(Strings)∧ Type ∈ Level_Names → ETYPE
}

3.4.4. Dimensiones (Dims).

Las dimensiones deben poder estructurarse en niveles, donde cada nivel debe tener una declaración en el conjunto de declaraciones del esquema.

La idea de cómo representar las dimensiones, es tomada también de [Cab97]. De esta forma, una dimensión es una tupla formada por un conjunto de niveles previamente declarados y un orden parcial con mínimo definido sobre los niveles. Toda dimensión, además, tendrá un indentificador.

Definición 3. Esquema de Dimensión.

D es un *Esquema de Dimensión* sobre las declaraciones **Decls**, si pertenece al siguiente conjunto:

Dimensions(Decls) ≡ { <DimName, L, Po> / DimName ∈ Strings
∧ L ∈ Set(Decls.Level_Names) ∧
∧ Po∈ Pos (L)
}

donde :

Pos (L) es el conjunto de los Ordenes Parciales sobre el conjunto **L**.

Esta especificación de la dimensión es similar a la propuesta por [CT97] como se mencionó anteriormente. La diferencia entre las dos propuestas en este punto, es que la familia de funciones de **DrillUp** que [CT97] propone, se considera parte de la instancia y no del esquema, ya que cada función debería ser dependiente de la instancia de cada nivel.

3.4.5. Relaciones Dimensionales (Rels).

Una *Relación Dimensional* debe representar el conjunto de todos los cubos posibles basados en determinadas dimensiones. Toda relación dimensional, tiene un nombre.

Definición 4. Esquema de Relación Dimensional.

R es un *Esquema de Relación Dimensional*, si pertenece al siguiente conjunto:

Relations(Dims) ≡ {<RelName,D > / RelName∈Strings ∧ D ∈Set(Dims) }

Puede parecer que las relaciones dimensionales son las abstracciones de [Cab97]. Sin embargo, esto no es así. Obsérvese que mientras las abstracciones propuestas por Cabbibo y Torlone pueden verse como relaciones entre niveles, las relaciones dimensionales son relaciones entre dimensiones completas.

3.4.6. Instancias.

Para comprender completamente el modelo, es necesario comprender además de los esquemas, las instancias de esos esquemas.

El enfoque tomado para la especificación de las instancias es que son objetos similares a los esquemas, de su mismo universo, pero que están en una determinada relación con los esquemas. Para lograr esto, se especifica una función llamada **Sinstance** que especifica cuál es el conjunto de las posibles instancias de cada estructura en el contexto de un esquema multidimensional.

Luego se presenta una función que dado un esquema multidimensional, describe las posibles instancias de la base de datos especificada por un esquema multidimensional dado.

3.4.6.1. Sinstance.

La función **Sinstance** se especifica mediante una macro como un conjunto paramétrico en la estructura y el esquema multidimensional. Para cada estructura y Esquema que recibe, se especifica una macro que al expandirse devuelve la expresión del conjunto de las posibles instancias para esa estructura según ese esquema. Sinstance se especifica como la unión de otros tres conjuntos paramétricos.

Definición 5. Instancia de un Esquema Multidimensional.

Si **E** es un esquema multidimensional y **e** una estructura en ese esquema, entonces **inst(e)** es una instancia de **e** si pertenece al siguiente conjunto:

Sinstance(E, e) ≡ Level_Sinst(E, e) ∪ Dim_Sinst(E, e) ∪ DimRel_Sinst(E, e)

Cada una de las macros **Sinst**, está definida de forma que si la estructura **e** no es de la categoría correcta (nivel, dimensión, etc), construyen el conjunto vacío.

3.4.6.2. Niveles.

Las instancias posibles de un nivel son subconjuntos finitos del tipo del nivel.

Definición 6. Instancia de un Nivel.

Si **E** es un esquema multidimensional y **l** es un esquema de nivel, entonces **inst(l)** es una instancia de **l** si pertenece al siguiente conjunto:

$$\textbf{Level\_SInst(E,l)} \equiv \{c \,/\, E \in \textbf{DimSch} \wedge l \in \textbf{E.Decls.Level\_names} \wedge c \in \textbf{Set(EType2Set(E.Decls.Type(l)))}\}$$

EType2Set es una función que interpreta un tipo, o sea, un elemento de **ETYPE** y devuelve el mismo tipo pero en términos del lenguaje de especificación.

3.4.6.3. Dimensiones.

Las instancias de una dimensión son las posibles jerarquías que se pueden construir “encadenando” objetos de un nivel con un objeto de cada uno de los niveles inmediatamente superiores (Fig. 5).

Para representar esto, la instancia de una dimensión en el modelo es una pareja formada por una función que dado el nivel devuelve la instancia del nivel y una familia de funciones de DrillUp que mapean los elementos de un nivel en los elementos de otro.

Definición 7. Instancia de una Dimensión.

Si **E** es un esquema multidimensional y **D** es un esquema de dimensión, entonces **inst(D)** es una instancia de **D** si pertenece al siguiente conjunto:

$$\textbf{Dim\_SInst(E,D)} \equiv \{ \langle il,du\rangle \,/\, E \in \textbf{DimSch} \wedge D \in \textbf{Dimensions(E.Dims)} \wedge \forall (l \in \textbf{D.L}).(il(l) \in \textbf{Level\_SInst(E,l)}) \wedge$$

$$(du \in (\{\langle l_1,l_2\rangle \in \textbf{D.Po}\} \rightarrow (il(l_1) \rightarrow il(l_2)))) \wedge$$

$$\forall (l \in \textbf{D.L}).\forall (l_1 \in \textbf{D.L}).\forall (l_2 \in \textbf{D.L}).$$

$$((\langle l,l_1\rangle \in \textbf{D.Po} \wedge \langle l_1,l_2\rangle \in \textbf{D.Po}) \rightarrow$$

$$\forall o \in l.(du(l_1,l_2)(du(l,l_1)(o)) = du(l,l_2)(o)))$$

$$\}$$

Notar que esta especificación garantiza la confluencia.

3.4.6.4. Relaciones Dimensionales.

Una instancia de una relación dimensional es el conjunto de los cubos que se pueden formar con niveles de las dimensiones que participan en la relación dimensional.

Un cubo es una función que mapea elementos de un producto cartesiano de instancias niveles en booleanos. A cada elemento de ese producto cartesiano se le llama *coordenada*. El valor de aplicar el cubo a una coordenada es verdadero cuando los datos de la coordenada están relacionados efectivamente en la realidad.

La definición de las instancias de una relación dimensional necesita de varias definiciones previas.

Definición 8. Coordenadas sobre un Conjunto de Niveles.

Si **E** es un esquema multidimensional y **L** es un conjunto de nombres de nivel, entonces **c** es una coordenada sobre ese conjunto de niveles si pertenece al siguiente conjunto:

$$\mathbf{Coord(E,L)} \equiv \{\mathbf{f} / \mathbf{E} \in \mathbf{DimSch} \wedge \mathbf{L} \subseteq \mathbf{E.Decls.Level\_Names} \wedge \mathbf{f} \in \{\mathbf{l} / \mathbf{l} \in \mathbf{L}\} \rightarrow \mathbf{Etype2Set(E.Decls.Types(l))}\}$$

Los posibles conjuntos de niveles que se pueden construir sobre una relación dimensional y que son de interés, deben tener al menos un nivel de cada dimensión. Estos conjuntos de niveles constituyen, en algún sentido, esquemas de cubos.

Definición 9. Esquemas de Cubos de una Relación Dimensional.

Si **E** es un esquema multidimensional y **RD** es un esquema de relación dimensional, entonces **c** es un esquema de cubo si pertenece al siguiente conjunto:

$$\mathbf{RD\_LevelSet(E,RD)} \equiv \{\mathbf{c} / \mathbf{E} \in \mathbf{DimSch} \wedge \mathbf{RD} \in \mathbf{RelDims} \wedge \mathbf{c} \in \mathbf{Set}(\{\mathbf{l} / \exists \mathbf{d} \in \mathbf{RD.l} \in \mathbf{d.L}\}) \wedge \forall(\mathbf{d} \in \mathbf{RD.D}).\exists(\mathbf{l} \in \mathbf{d.L}).\mathbf{l} \in \mathbf{c}\}$$

Como último paso, las instancias de las relaciones dimensionales son conjuntos de funciones que van de las coordenadas construidas sobre estos últimos conjuntos de niveles en los booleanos.

Definición 10. Instancia de una Relación Dimensional.

Si **E** es un esquema multidimensional y **RD** es un esquema de relación dimensional, entonces **inst(RD)** es una instancia de **RD** si pertenece al siguiente conjunto:

$$\mathbf{RD\_Sinst(E,RD) \equiv \{\ sf\ /\ E \in DimSch \wedge RD \in RelDims \wedge \exists (L \in RD\_LevelSet(E,RD)).(sf \in Set(Coord(E,L) \rightarrow Boolean))\}}$$

3.4.7. Instancias de una Base de Datos Multidimensional.

La instancia de un esquema de base de datos multidimensional está compuesta por las instancias de cada uno de sus componentes. Por lo tanto, debe tener una forma de determinar la instancia de cada componente en función de sus esquemas. Por esto, la instancia se ve como una tupla con funciones.

Definición 11. Instancia de un Esquema Multidimensional.

Si **E** es un esquema multidimensional, entonces **inst(E)** es una instancia de ese esquema si pertenece al siguiente conjunto:

$$\mathbf{Sinst\_Sch(E) \equiv \{<IL,ID,IR>\ /\ IL \in \{l/l \in E.Decls.Level\_names\} \rightarrow Sinstance(E,l) \wedge ID \in \{d/d \in E.Dims\} \rightarrow Sinstance(E,d) \wedge IR \in \{r/r \in E.RelDim\} \rightarrow Sinstance(E,r)\ \}}$$

3.5. Limitaciones de la Propuesta.

Las estructuras definidas en el caso de estudio de la sección 3.3, se pueden definir en términos de la formalización utilizando macro. De esta forma, asumiendo que los niveles han sido declarados, la dimensión *Vendedores* se puede definir como:

DVendedores ≡ ***<DimName**:vendedores,*

***L:**{Vendedor, Sucursales, Ciudades, Deptos, Regiones},*

***Po:**{ <Vendedor,Sucursales>,<Vendedor,Ciudades>,*

<Vendedor,Deptos>, <Vendedor,Regiones>,

<Sucursales,Ciudades>,<Sucursales,Deptos>,

<Sucursales,Regiones>,<Ciudades,Deptos>,

<Ciudades,Region>,<Depto,Region>}

>

Esta expresión representa una tupla con los elementos *DimName*, *L* y *Po*. El primer elemento debe ser el nombre de la dimensión como un string, el segundo un conjunto de nombres de niveles, que también son strings y el tercero una relación de orden entre los niveles del segundo conjunto. Esa relación de orden está representada como un conjunto de parejas.

Asumiendo que los niveles están declarados en el esquema, es fácil ver que se cumplen las condiciones por las que *Dvendedores* es una dimensión.

Esto debe tomarse como el mapeo del diagrama presentado en la Fig. 15 sobre el modelo formal.

De una forma similar, se podría especificar toda la estructura representada con los diagramas de la sección 3.3.2.

Sin embargo, la estructura que muestran los diagramas no refleja fielmente la realidad descripta en la sección 3.3.1. Esos diagramas no tienen en cuenta elementos como los siguientes:

- *Los vehículos utilitarios son camionetas, camiones o tractores, pero nunca son autos.*
- *Los vehículos familiares son autos o camionetas pero nunca son tractores o camiones.*
- *Los vendedores se especializan en vehículos de determinada categoría, o sea, o bien sólo venden utilitarios o bien sólo venden vehículos familiares pero no hay vendedores que vendan las dos categorías de vehículos ya que la empresa no lo permite de ninguna manera.*

Que la región sur está formada por los departamentos de Canelones, Montevideo y San José, no se puede especificar. Esto sería una propiedad de la función de *Drill-Up* de las instancias válidas para las dimensiones que manejen esa jerarquía.

Lo mismo sucede con el hecho de que los utilitarios son camiones o camionetas.

Que los vendedores se especializan en determinada categoría de vehículos, se tiene que especificar necesariamente en algún lugar de la estructura que relacionen vehículos con vendedores. Esto significa que esto debe ser una condición para las instancias de la relación dimensional *ventas*.

El problema es que para dar una buena representación de la realidad, no alcanza con las estructuras. Se necesita un lenguaje de restricciones y una estructura adecuada, que refleje la existencia de esas restricciones en el esquema.

En el siguiente capítulo se presenta una nueva versión de CMDM capaz de soportar un lenguaje de restricciones.

Capítulo 4. CMDM con Restricciones de Integridad.

4.1. Introducción.

A la hora de definir un lenguaje de restricciones para un modelo como CMDM se presentan dos opciones bastante claras:

- Definir un conjunto de restricciones ad-hoc, específicas para el modelo. Este sería el caso de las restricciones estructurales en el modelo Entidad-Relación.
- Definir un lenguaje de restricciones más general y luego, en algún sentido, programar las restricciones en este lenguaje.

Los dos casos son interesantes. Sin embargo, en el primero, el conjunto de restricciones que se pueden especificar es limitado y por lo tanto, también limita la expresividad del modelo. Tal como ocurre con el modelo Entidad-Relación, se hace necesario definir algún tipo de extensión para cubrir cada vez más restricciones o bien definir un lenguaje para las mismas.

En CMDM se sigue la segunda opción, o sea, se define un lenguaje general sobre el cual se pueden definir restricciones y también encapsularlas mediante macros. En diferentes dominios de aplicación, es posible que exista interés en diferente tipo de restricciones. Por lo tanto, es importante que los distintos usuarios puedan “personalizar” el lenguaje mediante el uso de macros adaptándolo a su realidad específica.

De esta forma, para extender el modelo con restricciones de integridad se necesita:

- Definir el lenguaje de restricciones.
- Modificar las estructuras de datos para que soporten las restricciones sobre ellas.

El lenguaje de restricciones que se define puede parecer complejo de usar por un usuario. Sin embargo, hay que tener en cuenta que el usuario que utilice el lenguaje de restricciones no será un usuario cualquiera, sino quien adapte el modelo a nuevo dominio de aplicación. Además, se parte de los siguientes dos supuestos:

- Se dispondrá de una herramienta CASE que manipule el modelo
- Las restricciones más usadas se podrán manipular en forma gráfica, ya sea dentro de la herramienta CASE o en forma manual.

En las siguientes secciones se presentan los fundamentos del lenguaje de restricciones y el análisis de algunas restricciones sobre el ejemplo anterior.

Luego se presenta la formalización del lenguaje de restricciones y por último la formalización del nuevo modelo.

4.2. Fundamentos del Lenguaje de Restricciones.

4.2.1. Introducción.

El lenguaje de restricciones que se propone es un lenguaje lógico de alto orden, en donde se permite la especificación de conjuntos por comprensión y extensión y los cuantificadores son siempre relativizados. El lenguaje es similar al lenguaje usado en la especificación del modelo, sólo que se utilizan las expresiones lógicas como lenguaje base[3]. También se propone hacer un uso intensivo de macros para mejorar la legibilidad y la escritura de las restricciones.

El lenguaje gráfico, se podrá extender con símbolos adecuados para la representación de ciertas restricciones que serán usadas en casi cualquier realidad.

Desde el punto de vista de una herramienta CASE, el hecho de que el lenguaje de restricciones sea lógico es relevante, ya que soportaría alguna noción de consistencia. Si el modelo que se define contiene un conjunto de restricciones inconsistente, entonces cualquier intento de cargar datos en un Data Warehouse que implemente ese modelo conceptual respetando las restricciones de integridad, será infructuoso y los programas de carga siempre desecharán todos los datos.

A continuación se presenta una reseña del lenguaje y una clasificación de las restricciones con el fin de ordenar la posterior presentación de las mismas sobre el caso de estudio.

4.2.2. Reseña del Lenguaje de Restricciones.

4.2.2.1. Generalidades del Lenguaje.

El lenguaje presenta los siguientes elementos:

- Operadores de comparación y pertenencia: $\in$, **<,>,**$\subseteq$**,=,…**
- Operadores básicos de formación de expresiones lógicas: $\wedge$, $\vee$, $\rightarrow$, $\neg$.
- Cuantificación sobre conjuntos: $\forall x \in S.\varphi$, $\exists x \in S.\varphi$
- Definición de Conjuntos por extensión y comprensión:***{a,b,c,…}, {c / φ }***
- Construcción y manipulación de pares: ***pair(a,b),fst(p),snd(p)***.
- Construcción de tuplas nominadas: **<a:1,b:5,c:7>,<a:3,b:5,c:10>**
- Funciones de acceso al esquema multidimensional: ***Levels, Dims, RelDims, Lpo, DrillUp, …***

La idea detrás del lenguaje, es que las interpretaciones del mismo se construyan en términos de los elementos de la base multidimensional.

[3] Mientras que en el lenguaje de especificación se hace a la inversa.

De esta forma, algunas de las constantes que aparecen en el modelo, se asocian con los nombres asociados en las estructuras que se definen en los esquemas. Cuando en la especificación del ejemplo aparece la expresión *deptos*, la misma será una constante que hace referencia a la instancia de algún elemento del esquema de nombre *deptos*, de acuerdo al contexto en que aparezca dicha expresión. De acuerdo con el ejemplo, seguramente será una referencia a la instancia del nivel "deptos" de la dimensión vendedores.

Además de las funciones de acceso al esquema, pueden aparecer funciones que corresponden a los tipos de los niveles. Si bien el tipo de un nivel podría ser cualquiera, en el ejemplo se asume que siempre son productos cartesianos (records).

Cada elemento del esquema, ya sea un nivel, una dimensión o una relación dimensional puede tener asociado un conjunto de restricciones.

4.2.2.2. Abreviaturas.

En el lenguaje de restricciones se pueden definir restricciones de identidad. Por ejemplo, en el nivel vendedor de la dimensión vendedores, es claro que no pueden existir diferentes vendedores con la misma cédula de identidad.

Esta restricción se puede escribir como:

$$\forall v_1 \in \textbf{Vendedores}.\ \forall v_2 \in \textbf{Vendedores}.(v_1.\textbf{cedula} = v_2.\textbf{cedula} \rightarrow v_1 = v_2)$$

Para simplificar la escritura, se admiten ciertas abreviaturas que se basan en que el elemento del esquema al que está asociada una restricción define un contexto para la misma. Este contexto es utilizado para eliminar la abreviaturas.

Por ejemplo, si la restricción anterior está asociada al nivel vendedores, se pueden omitir el conjunto de donde toman valores las variables obteniendo la siguiente expresión:

$$\forall v_1.\ \forall v_2.(v_1.\textbf{cedula} = v_2.\textbf{cedula} \rightarrow v_1 = v_2)$$

En este caso se asume que las variables toman valores en la instancia de la estructura asociada, que obviamente, deberá ser un conjunto.

Otro ejemplo de abreviaturas se puede ver en una restricción asociada a una dimensión. Si se omite la "cualificación" de un identificador, como por ejemplo DrillUp, entonces se asume que es una referencia al la función de DrillUp de esa dimensión.

4.2.2.3. Uso de Macros.

Para simplificar la escritura de las restricciones, es posible utilizar macros. De esta forma, la restricción que determinado atributo de un nivel determinado es una clave se puede describir con la siguiente macro:

$$\textbf{Level\_key(Att)} \equiv \forall v_1.\ \forall v_2.(v_1.\textbf{Att} = v_2.\textbf{Att} \rightarrow v_1 = v_2)$$

Para el caso del nivel *vendedor*, se aplicaría de la siguiente forma:

Level_key(cedula)

Nuevamente, esto solo permite la definición de un único atributo como clave.

Para permitir la definición de más de un atributo como clave, se puede usar la siguiente macro:

$$Level\_CompKey(Catt) \equiv \forall v_1.\forall v_2.((\forall att \in Catt.(v_1.att=v_2.att)) \rightarrow v_1=v_2)$$

Notar que las variables de los dos primeros cuantificadores toman valores de la instancia del nivel que corresponda pero, el tercero toma valores directamente del conjunto de atributos de ese nivel que se paso como parámetro (Catt) (Fig. 18).

Las definiciones de macros pueden aparecer en cualquier lugar en que pueda aparecer una restricción pero tiene que estar antes de su utilización.

4.2.3. Clasificación de las Restricciones.

Las restricciones se pueden clasificar en cuatro grandes categorías:

- **Intra-Nivel**. Se definen asociadas a un nivel dado. Son típicamente restricciones sobre conjuntos de datos.
- **Intra-Dimensión**. Se definen asociadas a una dimensión dada. En general se puede pensar que son condiciones que involucran a varios niveles de una jerarquía.
- **Intra-Relación Dimensional**. Se definen asociadas a una relación dimensional dada y en general, involucran a más de una dimensión o a niveles de más de una dimensión.
- **Del Esquema**. Son restricciones generales que se asocian al esquema y que pueden involucrar cualquier elemento de cualquier tipo del esquema, ya sean niveles, dimensiones, relaciones dimensionales, etc.

4.2.4.

Esta clasificación de las restricciones es un poco arbitraria. Para construirla se asume que cada estructura puede tener asociado un conjunto de restricciones.

En la siguiente sección, se presentan algunas restricciones típicas utilizando un lenguaje de alto orden y posibles representaciones gráficas. La presentación se pretende hacer en forma incremental y discutiendo diferentes tópicos a medida que se hacen necesarios.

4.3. Restricciones en el Caso de Estudio.

En la sección 3.5 se mostraron algunos inconvenientes para especificar ciertas restricciones sobre el modelo.

En esta sección se revisa el caso de estudio de la sección 3.3 agregando restricciones en forma gráfica y en un lenguaje de alto orden.

Se presentan las restricciones que parecen más obvias en cada tipo de restricción.

4.3.1. Restricciones sobre Niveles o Intra-Nivel.

Condiciones sobre Valores.

En CMDM es posible especificar condiciones sobre los elementos de cualquier conjunto, en particular, sobre los elementos de la instancia de un nivel. Por ejemplo, que la edad de todos los vendedores debe estar entre 18 y 66 años:

$$\forall v.(v.edad > 18 \wedge v.edad < 66)$$

Con este mecanismo se pueden escribir cualquier condición que sólo involucre elementos de un mismo nivel.

Desde el punto de vista gráfico, siempre se pueden asociar restricciones generales a cualquier elemento de un diagrama utilizando un rectángulo de puntas redondeadas y una flecha que indica a qué elemento se está asociando. (Fig. 18)

Identidad.

Otro tipo de restricción evidente sobre los elementos de un conjunto son las restricciones de identidad, ya mencionadas en las secciones 4.2.2.2 y 4.2.2.3.

A diferencia de las anteriores, estas restricciones se pueden expresar en forma gráfica sobre los diagramas. Para ello se sugiere subrayar el nombre del atributo que funcionará como identificador. (Fig. 18)

A nivel gráfico, se subrayaría todos los atributos que participan de la clave (en caso de no resolverlo como un atributo compuesto). Esta notación presenta el problema de no permitir la representación de otras claves alternativas en forma gráfica.

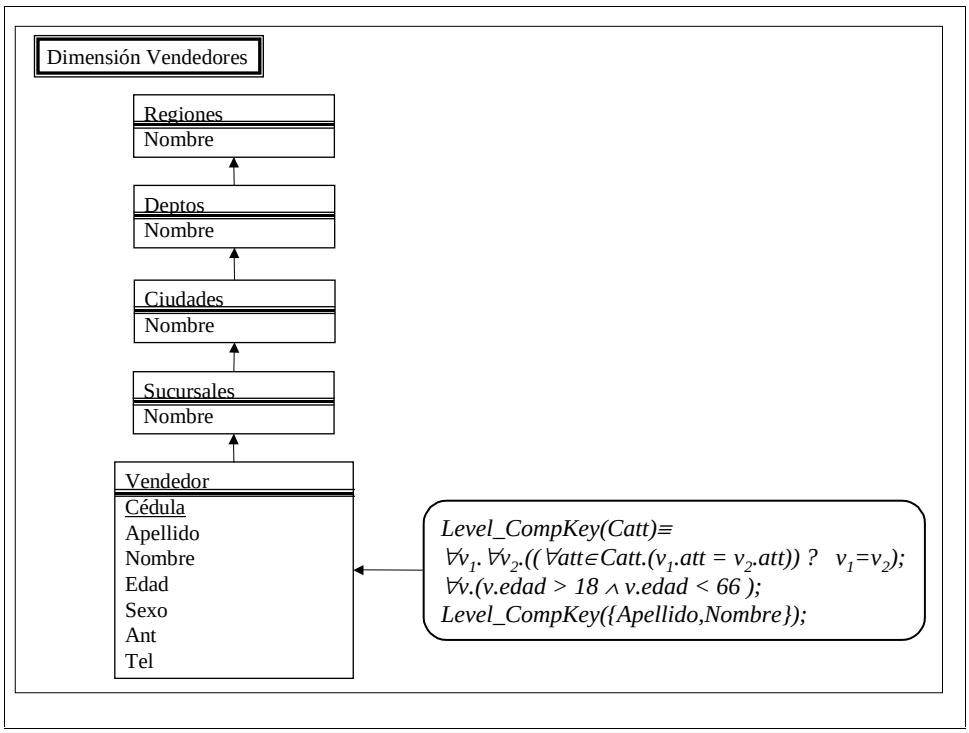

Fig. 18. Restricciones en el nivel Vendedor. Cédula es clave, además, los vendedores deben tener entre 18 y 66 años y Apellido y Nombre del vendedor constituyen una clave alternativa.

4.3.2. Intra-Dimensión.

Drill-Up.

Si las restricciones involucran a más de un nivel de una misma dimensión, en general, se van a asociar a esa dimensión y no a un nivel. Un caso particular son restricciones que terminan dando la semántica de la función de Drill-up.

Un caso extremo es definir explícitamente la función en determinados puntos. Un ejemplo claro es la asociación entre departamentos y regiones en la dimensión *Vendedores*:

$$\forall v \in \{e/e \in departamento \wedge e.nombre = \text{“Canelones”}\}.$$
$$\exists x \in region.$$
$$(DrillUp(departamento,region)(v) = x \wedge x.nombre = \text{”sur”})$$

Es claro que en la restricción anterior no se está asumiendo que *nombre* es clave en departamentos.

Para simplificar estas definiciones se puede definir la siguiente macro:

$$DefDeptoReg(NomDepto,NomRegion) \equiv \forall v \in \{e/e \in departamento \wedge e.nombre = NomDepto\}.(\exists x \in region.(DrillUp(departamento,region)(v) = x \wedge x.nombre = NomRegion))$$

Usando esta macro, las definiciones del drill-up de los departamentos a las regiones son de la siguiente forma:

DefDeptoReg("Canelones","Sur");

DefDeptoReg("Montevideo","Sur");

...

DefDeptoReg("Rio Negro","Oeste");

...

DefDeptoReg("Maldonado","Este");

...

DefDeptoReg("Artigas","Norte");

...

DefDeptoReg("Tacuarembo","Centro");

...

Resumiendo, con la utilización de este mecanismo se puede imponer casi cualquier restricción sobre la función de DrillUp. En principio, las primeras limitaciones estarán impuestas por los operadores básicos que se utilicen.

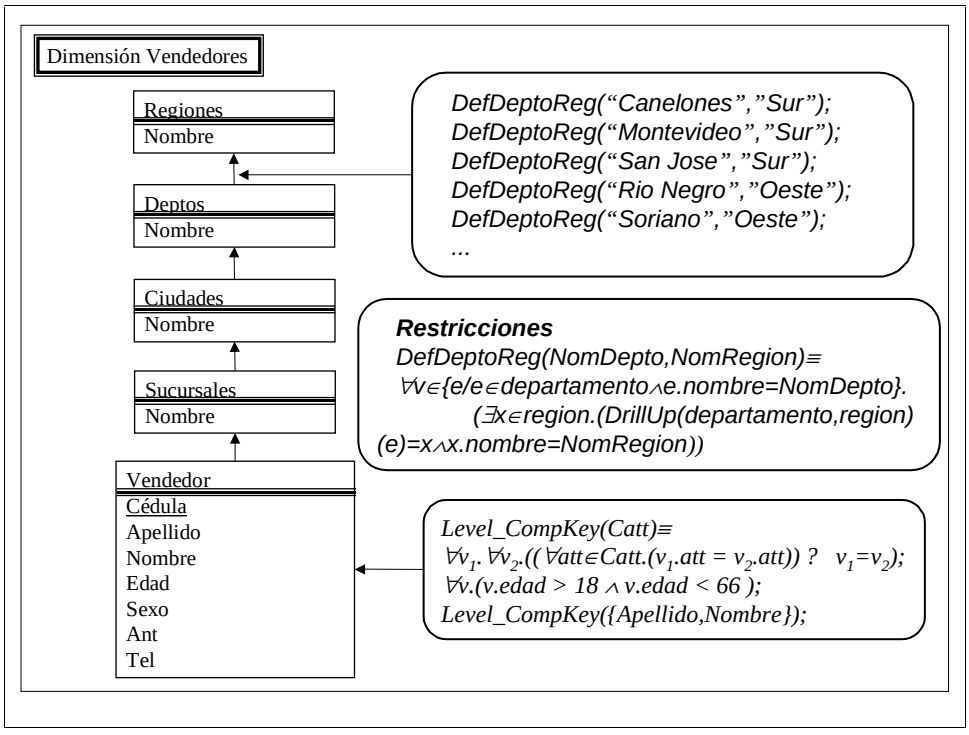

Fig. 19. Restricciones en la dimensión Vendedores. En el DrillUp de Deptos. a Región se imponen restricciones usando una macro definida a nivel de la dimensión.

Nuevamente, hay que tener en cuenta en dónde está declarada esta restricción. Al estar en relacionada con una dimensión, los nombres de nivel y la función de DrillUp son los correspondientes a esa dimensión. Tal es así que la macro puede colocarse en el mismo lugar en que van las restricciones generales de la dimensión (Fig. 19). Sin embargo, para dejar más claro el alcance, se puede asociar en otro rectángulo que apunta al link correspondiente.

Desde el punto de vista gráfico, lo que se está haciendo es construir una instancia particular para determinado “link” entre niveles (Fig. 20).

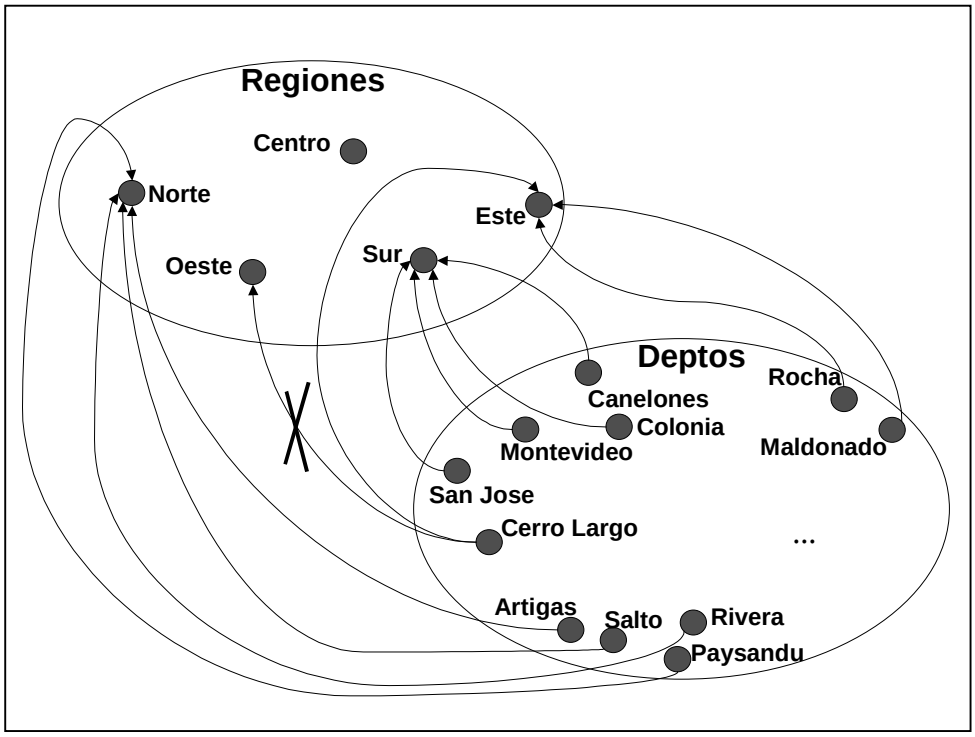

Fig. 20. Grafo de la función de DrillUp de Deptos a Regiones.

La relación de un nivel con el superior es N:1. Esto hace que si se declaran las regiones de todos los departamentos, el link tachado no sea válido.

Otras Restricciones.

También se pueden imponer las condiciones descriptas en la realidad sobre los vehículos y las categorías de vehículos:

- *Los vehículos utilitarios son camionetas, camiones o tractores, pero nunca son autos.*
- *Los vehículos familiares son autos o camionetas pero nunca son tractores o camiones.*

Cabe notar que estas condiciones relacionan niveles en jerarquías alternativas y no en una misma jerarquía.

Una forma de expresar estas condiciones es mediante la siguiente expresión:

$$\forall v \in Vehiculo.(\exists t \in Tipos.\exists g \in Categorias.(DrillUp(Vehiculo,Tipos)(v)=t \wedge DrillUp(Vehiculo,Categorias)(v)=g \wedge (g.cat=\text{"utilitario"} \rightarrow t.tipo \neq \text{"auto"}) \wedge (g.cat=\text{"familiar"} \rightarrow (t.tipo \in \{\text{"auto"},\text{"camioneta"}\})))$$

Estas restricciones deben ir asociadas a la dimensión (Fig. 21).

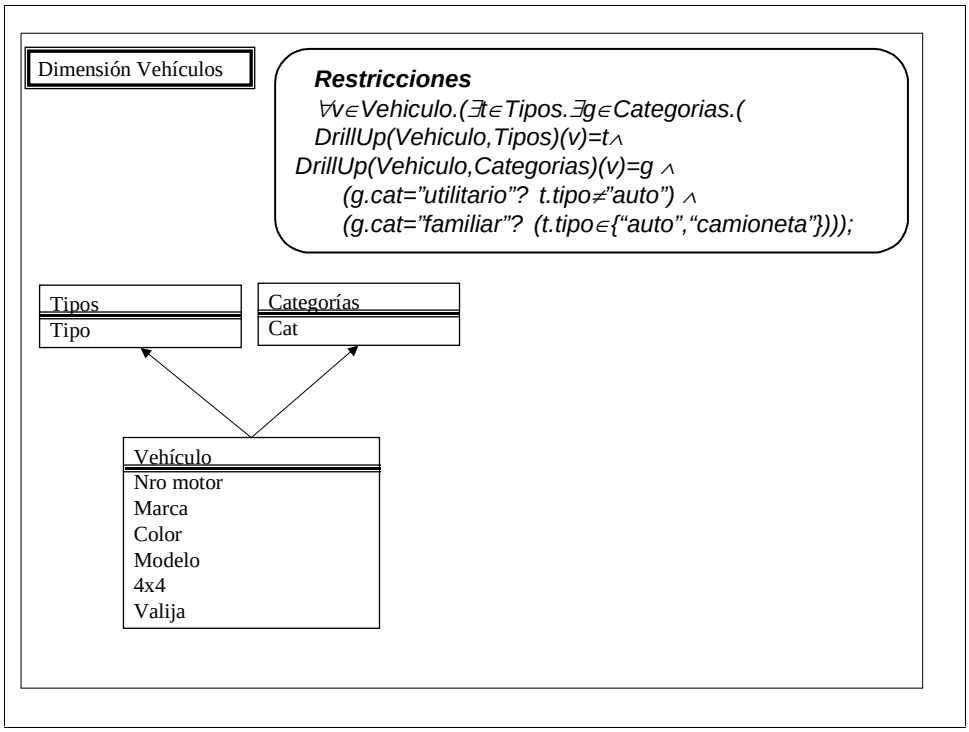

Fig. 21. Restricciones de la dimensión Vehículo. Como la restricción involucra varios niveles no relacionados, la restricción se asocia a la dimensión.

4.3.3. Intra-Relación Dimensional.

El primer punto para entender como escribir restricciones sobre las relaciones dimensionales es recordar qué es una instancia de una relación dimensional:

> *Una instancia de una relación dimensional es el conjunto de todos los cubos que se pueden definir tomando al menos un nivel de cada una de las dimensiones que participan de la relación.*

Por cubo, se entiende una función que va del producto cartesiano de los niveles que participan en el cubo en los booleanos. En cualquier cubo, deben estar todas las dimensiones de la relación dimensional, aunque no tienen porqué estar todos los niveles. Alcanza con que exista al menos un nivel por dimensión.

El conjunto de representaciones gráficas asociado a las restricciones sobre relaciones dimensionales puede ser considerado un lenguaje gráfico en sí mismo. Por este motivo, la representación gráfica de estas restricciones se presenta en la sección 4.3.4.

4.3.3.1. Cuantificación y Negación.

Una restricción posible en el ejemplo anterior, es que sólo interesan aquellos cubos que tienen como medida el nivel *venta* de la dimensión *ventas*.

Para esto se puede asociar la siguiente restricción a la relación dimensional *Transacciones*:

$$\forall c.Medida(c,Ventas.Venta)$$

Un cubo es una función del producto cartesiano de todos los niveles que participan en su definición en los booleanos. El resultado del cubo debería ser verdadero cuando la coordenada que se utiliza es efectivamente una coordenada definida para ese cubo. En este contexto, que determinado nivel es una medida significa que hay una relación funcional con recorrido en este nivel y con dominio los otros niveles.

La macro Medida, se puede definir de la siguiente forma:

$$Medida(c,l) \equiv \neg(\exists f \in CoordT(c).\exists f_1 \in CoordT(c).(\forall l_1 \in Dom(c).(l \neq l_1 \rightarrow f(l_1)=f_1(l_1)) \wedge f(l) \neq f_1(l)))$$

Esta macro indica que dado un cubo *c* y un nivel *l*, no pueden existir dos coordenadas en el cubo que tengan iguales valores en los otros niveles y valor distinto en *l*.

En la definición de la macro *Medida* aparecieron dos expresiones que no están definidas anteriormente: *CoordT* y *Dom*.

La macro *CoordT* especifica el conjunto de las coordenadas que tienen como resultado el valor *true*. La expresión *Dom(c)* denota el esquema del cubo *c,* o sea, el conjunto de los niveles que participan de las coordenadas del cubo.

La macro *CoordT* se puede definir de la siguiente forma:

$$CoordT(c) \equiv \{f \,/\, f \in Coord(c) \wedge c(f)=true\}$$

$$Coord(c) \equiv \{f \,/\, \forall l \in Dom(c).\exists v \in l.f(l)=v\}$$

Un punto a observar en las definiciones anteriores es que las coordenadas son funciones que mapean los niveles[4] en elementos de sus instancias. Esto no es otra cosa que ver un producto cartesiano como una función.

Notar que con estas macros se pueden pedir otras restricciones más débiles, como por ejemplo que la medida sea algún nivel de la dimensión ventas, sin especificar cuál:

$$\exists l \in Levels(Ventas).Medida(c,l)$$

O bien, que no puede haber ningún cubo que tenga el nivel venta de ventas como medida:

$$\neg \exists c.(Medida(c,Ventas.Venta))$$

Las condiciones pueden ser tan complejas como sea necesario.

4.3.3.2. Condicionamiento de las Coordenadas.

Realmente, no hay diferencia entre coordenadas propiamente dichas y medidas: la medida es una coordenada más y esa filosofía queda expresada en las macros anteriores.

De esta forma, no hay diferencia en imponer condiciones sobre las dimensiones o las medidas de un cubo.

Por ejemplo, la siguiente condición exige la utilización de casi todo el conjunto de operadores presentado anteriormente:

> *Los vendedores se especializan en vehículos de determinada categoría, o sea, o bien sólo venden utilitarios o bien sólo venden vehículos familiares pero no hay vendedores que vendan las dos categorías de vehículos ya que la empresa no lo permite de ninguna manera.*

Esto significa que cualquier venta que realice un vendedor, debe ser dentro de su especialidad y que no hay excepciones de ningún tipo.

Para expresar esto hay que indicar que sobre los cubos que tengan ventas como medida, la categoría de los vehículos que vende un vendedor siempre es la misma[5]:

[4] En realidad debiera ser la identificación de los niveles, pero para simplificar la notación se utiliza el nivel tanto cuando debe aparecer la identificación como cuando debe aparecer la instancia. En todo caso, eso se puede tratar en el lenguaje observando el tipo de la expresión ya que la identificación es de tipo *i* mientras que la instancia es de tipo *set(i)*.

[5] La *V* que aparece calificando algunas expresiones es una abreviatura para *Vendedores*.

$$\forall c.((Medida(c,Ventas.Venta) \wedge V.Vendedor \in Dom(c) \wedge Vehículos.Vehículo \in Dom(c) \rightarrow \forall t_1 \in CoordT(c).\ \forall t_2 \in CoordT(c).\ (t_1(V.Vendedor)=t_2(V.Vendedor) \rightarrow Vehículos.DrillUp(Vehículo,Categoría)(t_1(Vehículo))= Vehículos.DrillUp(Vehículo,Categoría)(t_2(Vehículo))$$

$$)$$

1.1.1.1.)

En la restricción, se pide que para todos los cubos *c*, si ese cubo tiene medida *Ventas.Venta* y *Vendedores.Vendedor* está en el dominio de *c*, entonces cualquier par de tuplas que tengan el mismo vendedor, deben tener asociado la misma categoría de vehículo. Para obtener la categoría de vehículo, es necesario hacer un DrillUp hasta el nivel adecuado.

4.3.3.3. Expresando la Estructura de los Cubos y RollUp's Pre-hechos.

Con una nueva macro se pueden imponer condiciones sobre la estructura de los cubos, o mejor dicho, se pueden especificar los esquemas de los cubos.

$$Cubo(c,L,l) \equiv (Dom(c)=L \cup \{l\}) \wedge Medida(c,l)$$

Claramente, hay tres conjuntos de restricciones que pueden reconocerse como útiles de antemano:

- Que todos los cubos de la instancia de la relación dimensional tienen determinada estructura.

$$\forall c.Cubo(c,L,l)$$

- Que hay un cubo en instancia de la relación dimensional que tiene determinada estructura.

$$\exists c.Cubo(c,L,l)$$

- Que no hay ningún cubo en la relación dimensional que tenga determinada estructura.

$$\neg \exists c.Cubo(c,L,l)$$

Estas restricciones dan origen a un conjunto de estructuras gráficas que simplifican la definición de los cubos. Estas estructuras se presentan en la sección 4.3.4.

A pesar de que las condiciones anteriores ya bastarían para justificar la definición de las macros anteriores, aún tienen más aplicaciones. En base a estas macros se pueden expresar algunas condiciones interesantes en las que intervienen más de un cubo de la instancia de la relación dimensional.

Dados dos cubos *c* y c_1, la siguiente macro especifica cuando el cubo c_1 es el resultado del rollup por la dimension *dim* desde el nivel *linf* al nivel *lsup* usando el operador *o* sobre el nivel *m* de la dimension *mdim*, asumiendo (o más bien chequeando) que *mdim.m* es una medida de c y que el nivel resultante es el nivel *mup* de *mdim*:

$$
\begin{aligned}
&\mathbf{IsRollUpOne(c,dim,linf,sup,mdim,m,mup,o,c_1)} \equiv \\
&\quad \mathbf{Medida(c,mdim.m)} \wedge \mathbf{dim.linf} \in \mathbf{Dom(c)} \wedge \\
&\quad \mathbf{pair(dim.linf,dim.lsup)} \in \mathbf{dim.PO} \wedge \\
&\quad \mathbf{Medida(c_1,mdim.mup)} \wedge \\
&\quad \mathbf{Dom(c_1)=Dom(c)\{dim.linf,mdim.m\} \cup \{dim.lsup,mdim.mup\}} \wedge \\
&\quad \mathbf{pair(mdim.m,mdim.mup)} \in \mathbf{mdim.PO} \wedge \\
&\quad \forall \mathbf{t} \in \mathbf{Coordt(c)}.\exists \mathbf{t_1} \in \mathbf{Coordt(c_1)}.\ \forall \mathbf{l} \in \mathbf{Dom(c)}.((\mathbf{l=dim.linf} \wedge \\
&\qquad \mathbf{t_1(dim.lsup)= dim.DrillUp(linf,lsup)(t(dim.linf)))} \vee \\
&\qquad \mathbf{(l=mdim.m} \wedge \mathbf{t_1(mdim.mup)=} \\
&\qquad \mathbf{o(\{v/\exists t_2 \in Coordt(c)} \wedge \mathbf{v=t_2(mdim.m)} \wedge \mathbf{t_2(dim.linf)=t(dim.linf)\})} \\
&\qquad \mathbf{)} \vee \mathbf{t(l)=t_1(l)} \\
&\quad \mathbf{)}
\end{aligned}
$$

La macro anterior es una conjunción de siete fórmulas en donde las seis primeras expresan que:

- *mdim.m* es una medida de *c*
- *dim.linf* es un nivel del esquema de *c*
- *lsup* es un nivel superior a *linf* según el orden establecido en *dim*
- *mdim.mup* es una medida de c_1
- Los niveles de c_1 son los mismos están en *c* excepto que en vez de *linf* está *lsup* y en vez de *m* está *mup*.
- *mup* es un nivel superior a *m* según el orden establecido por *mdim*

La última expresa que para cualquier coordenada *t* de c, debe haber una coordenada t_1 en c_1 que cumpla:

- El valor de *dim.lsup* en t_1 es el resultado del *DrillUp* de *linf* a *lsup* según la jerarquía de *dim* aplicado al valor de *dim.linf* en *t*.
- El valor de *mdim.mup* en t_1 es el resultado del aplicar la operación *o* sobre los valores del nivel *mdim.m* a aquellas coordenadas de *c* que tienen el mismo valor en *dim.linf* que la coordenada *t*.
- En cualquier otro caso, el valor de las coordenadas es el mismo.

Estas son las condiciones que definen que un cubo es el resultado de un rollup sobre determinada dimensión de otro cubo y aplicando determinado operador a una medida del cubo original.

Notar que los operadores de consolidación que se manejan como el parámetro *o*, debe devolver un elemento de un nivel dado. Por eso es que se asume que hay una función por cada nivel cuyo nombre es el nombre del nivel precedido por los caracteres *mk_* que construye un elemento del tipo del nivel cuando se le pasan los parámetros adecuados. En la sección 4.3.4.5 quedará más clara la utilidad de estas operaciones.

Sin embargo, para especificar el DrillUp simultáneo en varias dimensiones con esta macro no alcanza. Para lograr esto, la macro debe recibir un conjunto de ternas, dimensión, nivel inferior y nivel superior:

$$
\begin{aligned}
&IsRollUpSim(c,SL,mdim,m,mup,o,c_1) \equiv \\
&\quad Medida(c,mdim.m) \wedge \\
&\quad Medida(c_1,mdim.mup) \wedge \\
&\quad \forall e \in SL.(\ e(dim).e(linf) \in Dom(c) \wedge \\
&\qquad pair(e(dim).e(linf),e(dim).e(lsup)) \in e(dim).PO \\
&\quad) \wedge \\
&\quad Dom(c_1)=(Dom(c)-\{d.l/\exists e \in SL.(e(dim)=d \wedge e(linf)=l)\}-\{mdim.m\}) \\
&\qquad \cup(\{d.l/\exists e \in SL.(e(dim)=d \wedge e(lsup)=l\} \cup \{mdim.mup\}) \wedge \\
&\quad \forall t \in Coordt(c).\exists t_1 \in Coordt(c_1). \\
&\qquad \forall l \in Dom(c).(\\
&\qquad\quad \exists e \in SL.(l=e(dim).e(linf) \wedge \\
&\qquad\qquad t_1(e(dim).e(lsup))= \\
&\qquad\qquad e(dim).DrillUp(e(linf),e(lsup))(t(e(dim).e(linf))) \\
&\qquad\qquad) \vee \\
&\qquad\quad (\ l=mdim.m \wedge \\
&\qquad\qquad t_1(mdim.mup)=o(\{v/\exists t_2 \in Coordt(c) \wedge v=t_2(mdim.m) \wedge \\
&\qquad\qquad \forall e \in SL.t_2(e(dim).e(linf))=t(e(dim).e(linf)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \}) \\
&\qquad\quad) \vee \\
&\qquad\quad (t(l)=t_1(l)) \\
&\qquad)
\end{aligned}
$$

Esta macro no es más que una extensión de la macro anterior, en la que se supone que el conjunto *SL* contiene ternas *<dim,linf,lsup>* en donde *dim*,*linf* y *lsup* son nombres de atributos. Las condiciones que en *IsRollUpOne* se imponían sobre *linf* y *lsup*, ahora se imponen sobre los elementos de *SL* por lo que están cuantificadas sobre dicho conjunto.

4.3.3.4. Resumibilidad.

Este tipo de condiciones indican que operaciones de RollUp se pueden realizar y cuáles no sobre determinada medida en un determinado cubo.

Para especificar que se puede realizar una determinada operación de rollup sobre determinado cubo, utilizar *IsRollUpOne* cuantificando existencialmente algunos parámetros:

$$\exists c.\exists l \in Dom(c).$$
$$IsRollUpOne(cventa,vendedores,vendedor,$$
$$ciudad,ventas,venta,l,PromPrecio,c)$$

Esta restricción está indicando que sobre el cubo cventa, al realizar el DrillUp de *vendedor* a *ciudad* sobre la dimensión *Vendedores*, existe un cubo *c* tal que es el resultado de aplicar el operador *PromPrecio* sobre los agrupamientos resultantes del DrillUp. Notar que se está indicando también que debe existir un nivel *l* que debería (según la macro *IsRollUpOne*) ser un nivel superior a ventas, pero que de acuerdo a la definición de la dimensión, no existe. Este nivel debe ser agregado en el esquema de la dimensión correspondiente.

4.3.4. Algunas Nociones sobre un Lenguaje Gráfico para Restricciones Intra-Relación Dimensional.

La mayoría de las restricciones Intra-Relación Dimensional, limitan el conjunto de cubos que pueden pertenecer a la instancia de la relación.

Las macros definidas anteriormente, facilitan la definición de estas restricciones, sin embargo, se puede extender el lenguaje gráfico para cubrir las restricciones más comunes.

4.3.4.1. Cubos.

La mayoría de las restricciones sobre relaciones dimensionales se pueden escribir como:

- *En la relación, Existe un cubo que cumple...*
- *Todo cubo de la relación cumple que ...*
- *No hay ningún cubo en la relación que cumpla que ...*

Esta visión de las restricciones sobre una relación dimensional, permite la construcción de una representación gráfica bastante intuitiva. Esta representación se basa en un grafo en cual un nodo central “cúbico” (ver figuras) se conecta con una representación de cada uno de los niveles que participan en ese cubo. El nodo central, indica que tipo de cuantificación tiene la restricción.

En la Fig. 22 se puede ver una restricción existencial que indica que en toda instancia de la relación dimensional *Transacciones*, debe existir al menos un cubo con los niveles más bajos de agregación de los dimensiones que participan en la relación dimensional. La flecha sobre el nivel *ventas.venta* indica que dicho nivel es considerado como la medida.

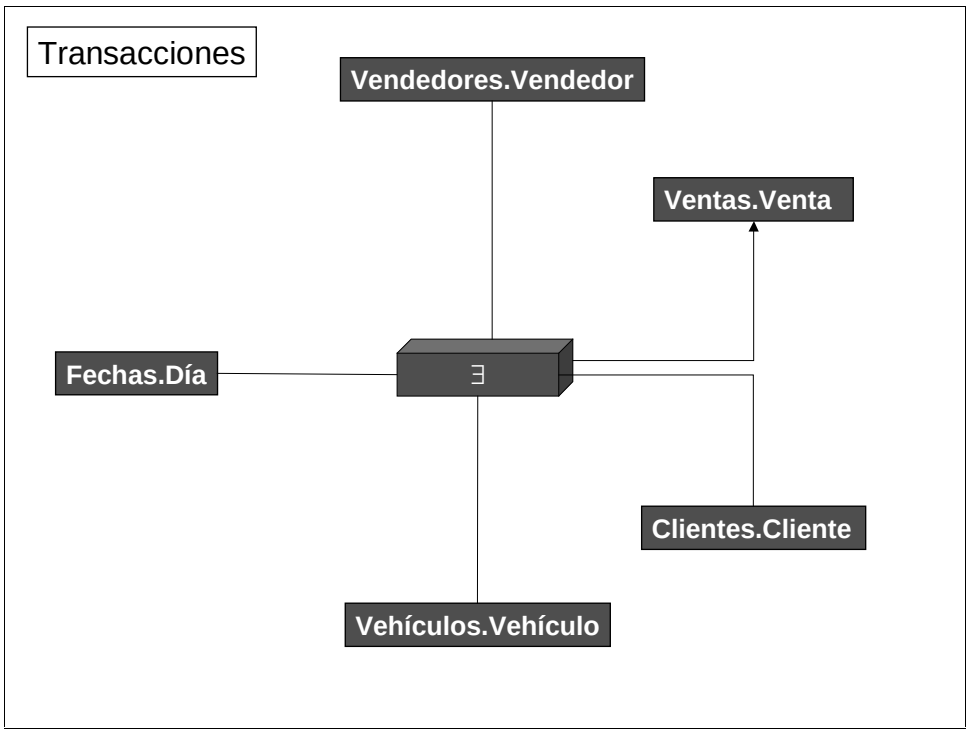

Fig. 22. Representación Gráfica de una Restricción Existencial sobre la Relación Dimensional Transacciones.

Esta especificación gráfica, tiene asociado el siguiente código en el lenguaje de restricciones:

```
∃c.Cubo(  c,
          { Clientes.Cliente,
            Vehículos.Vehículo,
            Fechas.día,
            Vendedores.Vendedor
          },
          Ventas.Venta
    )
```

Si el nodo central tuviera el símbolo ∀ en vez de ∃, el código sería:

```
∀c.Cubo(   c,
                { Clientes.Cliente,
                  Vehículos.Vehículo,
                  Fechas.día,
                  Vendedores.Vendedor
                },
                Ventas.Venta
        )
```

Si el símbolo del nodo central fuera ¬, entonces el código sería:

```
¬∃c.Cubo(   c,
                { Clientes.Cliente,
                  Vehículos.Vehículo,
                  Fechas.día,
                  Vendedores.Vendedor
                },
                  Ventas.Venta
        )
```

Si en el lugar del símbolo lógico en el nodo central aparece un identificador, por ejemplo *cventas,* el mismo es interpretado como una constante y el código asociado es el siguiente:

```
Cubo(cventas,
              { Clientes.Cliente,
                Vehículos.Vehículo,
                Fechas.día,
                Vendedores.Vendedor},
                Ventas.Venta
        )
```

4.3.4.2. Dimensionalidad Genérica.

Las medidas, están indicadas con una punta de flecha. La dimensionalidad genérica, estará expresada por la inexistencia de dicha marca (Fig. 23).

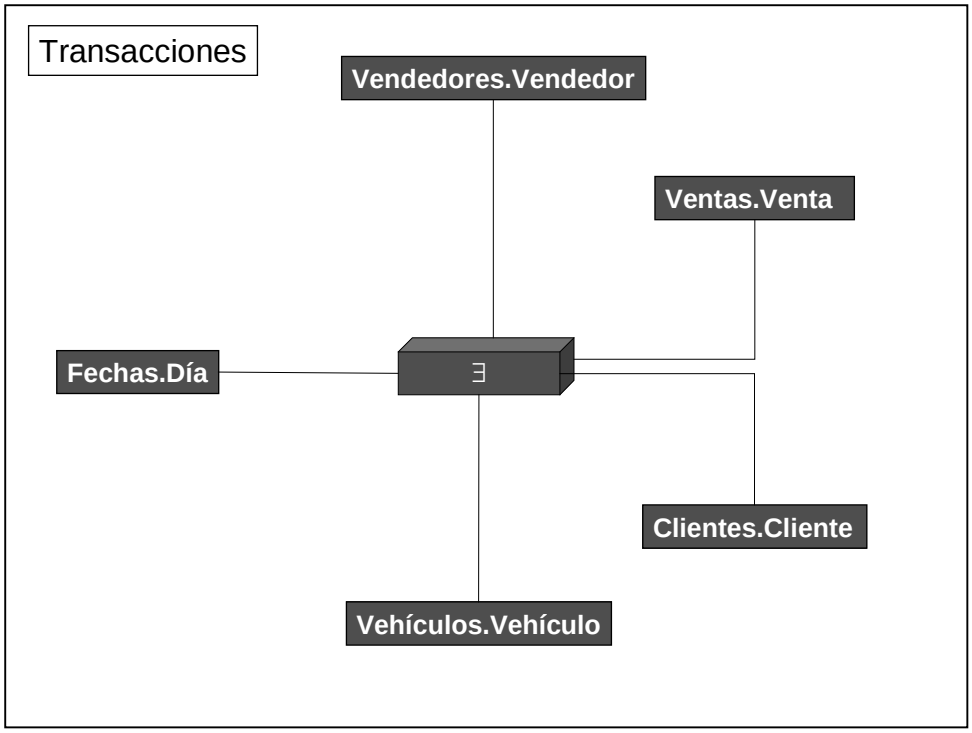

Fig. 23. Restricción Existencial con Dimensionalidad Genérica, no hay ninguna diferencia entre ninguna de las dimensiones participantes.

Lo que expresa esta restricción es que existe al menos un cubo en la instancia de la relación dimensional con determinado dominio:

$$\exists c.(Dom(c)= \{ Clientes.Cliente, Vehículos.Vehículo, Fechas.día, Vendedores.Vendedor, Ventas.Venta \})$$

Obviamente, se pueden escribir en forma gráfica restricciones similares a ésta pero con cuantificación universal, negación o introducción de constantes.

4.3.4.3. Dimensionalidad Acotada y Múltiple.

La existencia o no de una medida conocida son los casos extremos de un conjunto de situaciones. En realidad, se podría expresar que debe existir un cubo en el cual más de un nivel puede funcionar como medida simultáneamente (Fig. 24).

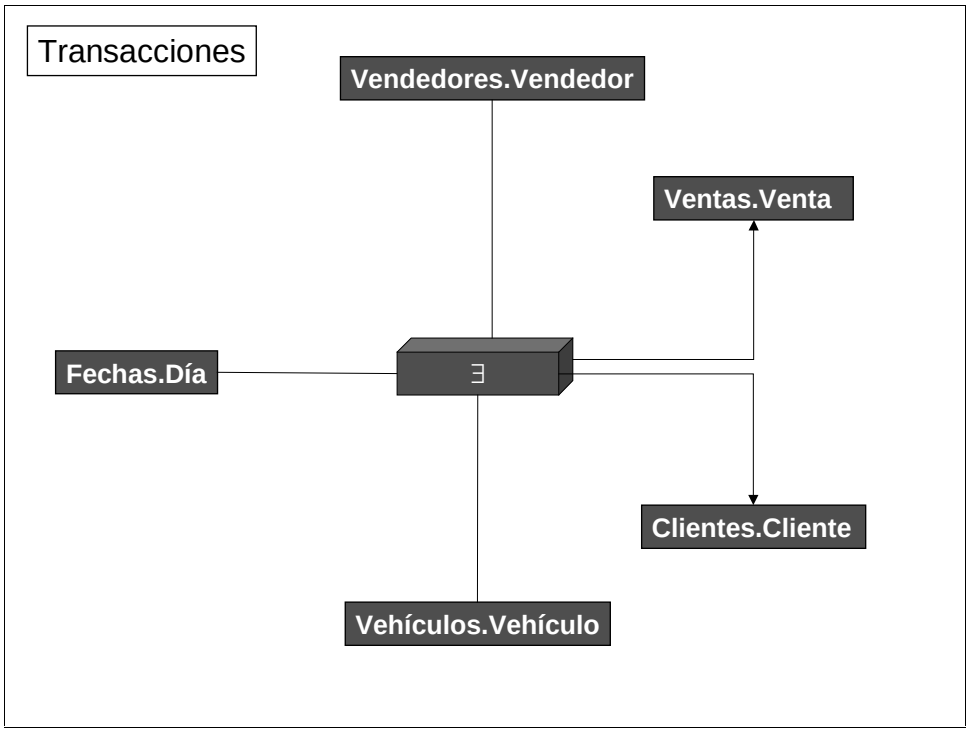

Fig. 24. Ventas.Venta y Clientes.Cliente son medidas a la vez.

El código que corresponde a la figura anterior es el siguiente:

```
∃c.(Dom(c)= { Clientes.Cliente,
              Vehículos.Vehículo,
              Fechas.día,
              Vendedores.Vendedor,
              Ventas.Venta
            }∧
    Medida(c,Ventas.Venta)∧
    Medida(c,Clientes.Cliente)
)
```

Recordar que la macro medida expresa que la relación entre los valores de un cubo es funcional con respecto al nivel distinguido. Por esto, no hay problema con que más de un nivel sea una medida a la vez en el cubo.

También se puede expresar que varios niveles que pueden ser la medida de determinado cubo, aunque no necesariamente en forma simultánea (Fig. 25).

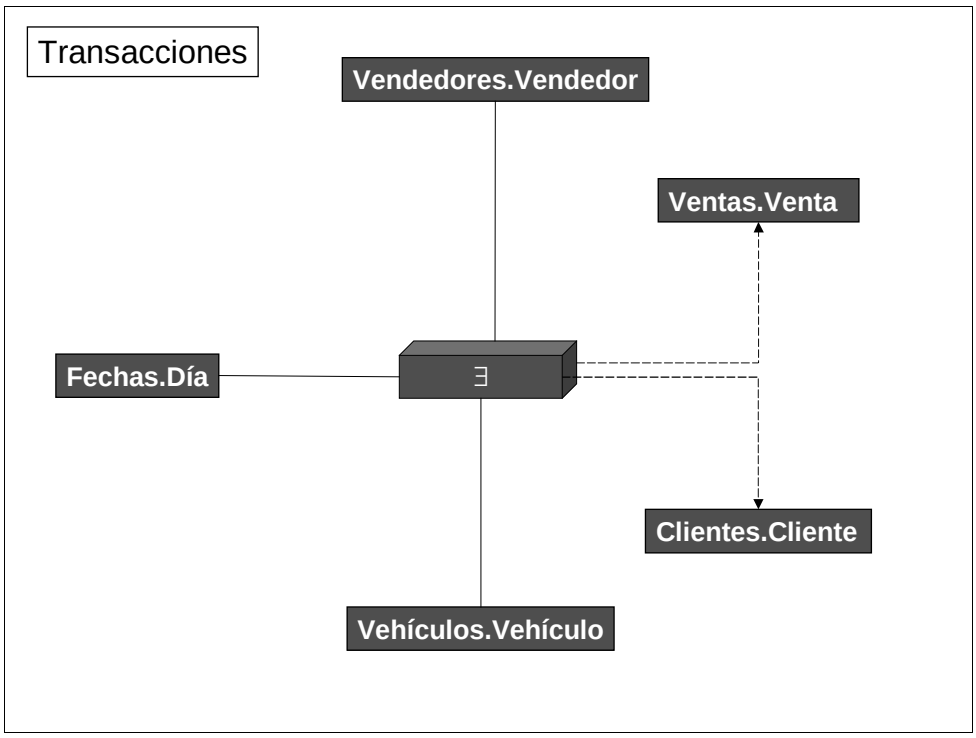

Fig. 25. La medida del cubo, es ventas.venta o es clientes.cliente (o los dos).

El código asociado a la restricción anterior es:

$$\exists c.(\ (Dom(c)=\{\ Clientes.Cliente,\ Vehículos.Vehículo,\ Fechas.día,\ Vendedores.Vendedor,\ Ventas.Venta\ \})\wedge (\ Medida(c,Ventas.Venta)\vee Medida(c,Clientes.Cliente))\ )$$

Aplicando equivalencias lógicas, se puede ver que esto no es más que una forma abreviada de escribir que existe un cubo con una de las medidas o existe un cubo con la otra:

$$\exists c.((Dom(c)=\{\ Clientes.Cliente,\ Vehículos.Vehículo,\ Fechas.día,\ Vendedores.Vendedor,\ Ventas.Venta\})\wedge Medida(c,Ventas.Venta)\ )\vee \exists c.((Dom(c)=\{Clientes.Cliente,\ Vehículos.Vehículo,\ Fechas.día,\ Vendedores.Vendedor,\ Ventas.Venta\})\wedge Medida(c,Clientes.Cliente)$$

Con estas notaciones es posible expresar gráficamente condiciones complejas, por ejemplo, usando las dos notaciones a la vez (Fig. 26).

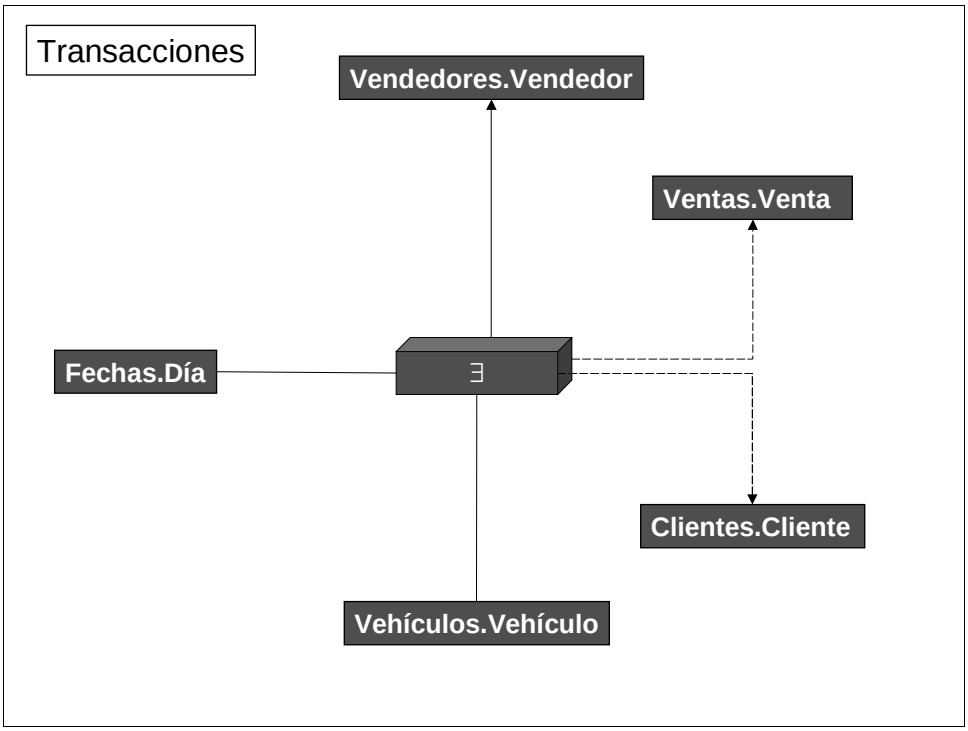

Fig. 26. Vendedor.Vendedores es una medida y Ventas.Venta o Clientes.Cliente son medidas también.

El código correspondiente a la restricción anterior es:

$$\exists c.((Dom(c)= \{ Clientes.Cliente, Vehículos.Vehículo, Fechas.día, Vendedores.Vendedor, Ventas.Venta\}) \wedge Medida(c,Vendedores.Vendedor) \wedge (Medida(c,Ventas.Venta) \vee Medida(c,Clientes.Cliente)))$$

Cuando se usan ambas notaciones, siempre se genera la conjunción de, la conjunción de las medidas marcadas con las líneas completas con, la disyunción de las medidas marcadas con flechas punteadas. En otras palabras, la estructura de la fórmula será:

$$<cuantificador>c. (Dom(c)=<Conjunto\ de\ los\ niveles\ participantes> \wedge Medida(c,<nivel\ con\ flecha\ completa>) \wedge Medida(c,<nivel\ con\ flecha\ completa>) \wedge ... \wedge (Medida(c,<nivel\ con\ linea\ punteada>) \vee Medida(c,<nivel\ con\ linea\ punteada>) \vee ....))$$

Hay que tener cuidado con las expresiones complejas. En la Fig. 25 se presenta un caso que es equivalente a especificar que existe al menos uno de dos cubos, uno con una medida y otro con otra. Si simplemente se cambia el cuantificador existencial por uno universal, entonces se está especificando que todo cubo o tiene una medida o tiene la otra y eso no es equivalente a nada particular.

Otro caso de dimensionalidad limitada es que todos los cubos deban tener la medida en determinada dimensión. Eso se representa directamente en el diagrama de la relación dimensional, poniendo una punta de flecha en determinada dimensión (Fig. 27).

El código correspondiente a la punta de flecha sobre la dimensión *ventas* en la relación dimensional *Transacciones* es:

$$\forall c.\exists l \in Levels(Ventas).Medida(c,l)$$

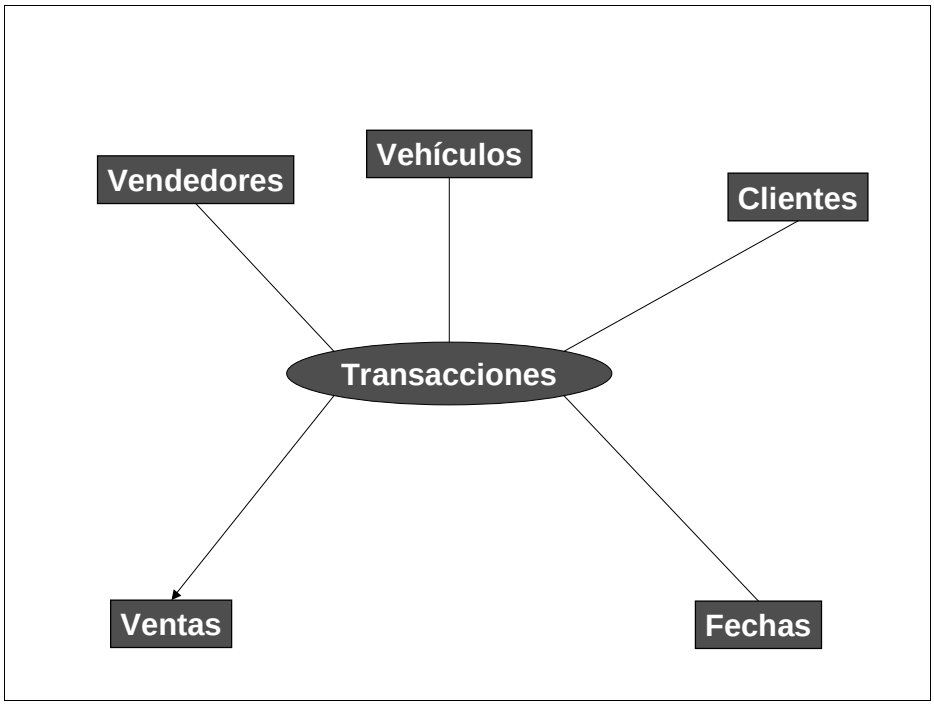

Fig. 27. Todos los cubos de la relación dimensional deben tener la medida en la dimensión ventas.

4.3.4.4. Restricciones Generales.

En caso de que para determinada restricción no exista una notación gráfica adecuada, siempre es posible escribirla en el lenguaje de restricciones y asociarla a un contexto determinado.

Si se asocian en el diagrama de la relación dimensional se interpreta que esa condición debe ser cumplida por todos los cubos de la instancia (Fig. 28).

El código para el ejemplo presentado en la figura consta de la macro y una invocación cuantificada universalmente:

$$
\begin{aligned}
&VendedorFcat(c) \equiv (\\
&\quad (Medida(c, Ventas.Venta) \wedge V.Vendedor \in Dom(c) \wedge \\
&\qquad Vehículos.Vehículo \in Dom(c) \\
&\quad) \rightarrow \forall t_1 \in CoordT(c). \\
&\qquad\qquad \forall t_2 \in CoordT(c).(\\
&\qquad\qquad\quad t_1(V.Vendedor) = t_2(V.Vendedor) \rightarrow \\
&\quad Vehículos.DrillUp(Vehículo, Categoría)(t_1(Vehículo)) = \\
&\quad Vehículos.DrillUp(Vehículo, Categoría)(t_2(Vehículo)) \\
&\qquad\qquad) \\
&) \\
&\forall c.VendedorFcat(c)
\end{aligned}
$$

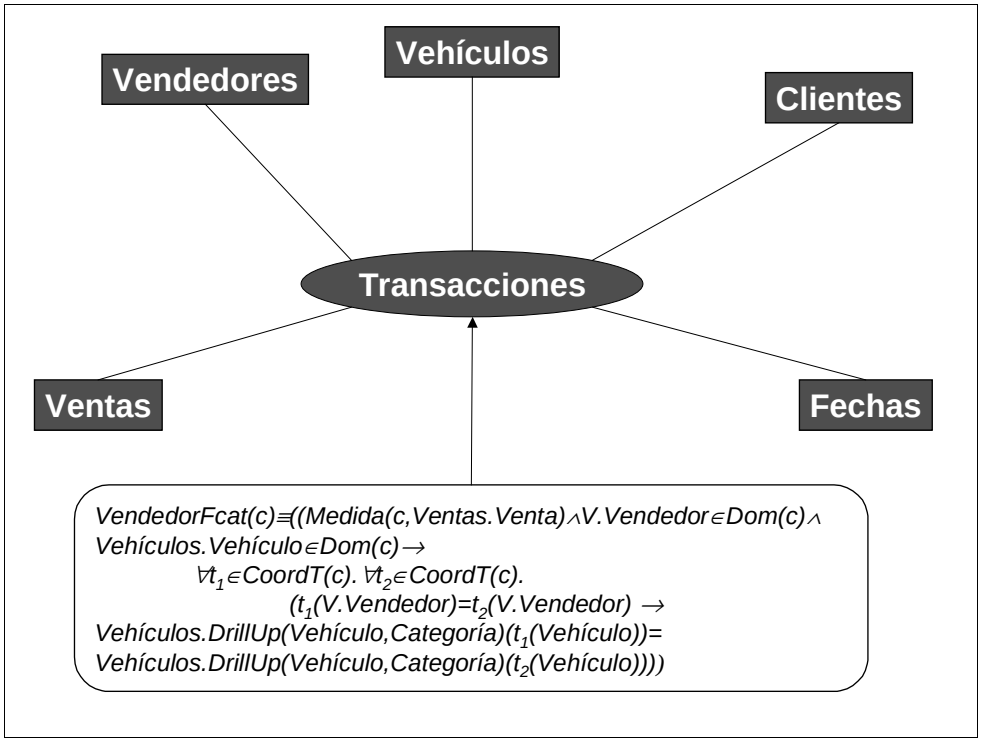

Fig. 28. Todos los cubos de la instancia de Transacciones deben respetar que un vendedor sólo vende vehículos de determinada categoría.

Si se asocia a un cubo determinado, se agrega como una condición más dentro de la expresión cuantificada(Fig. 29).

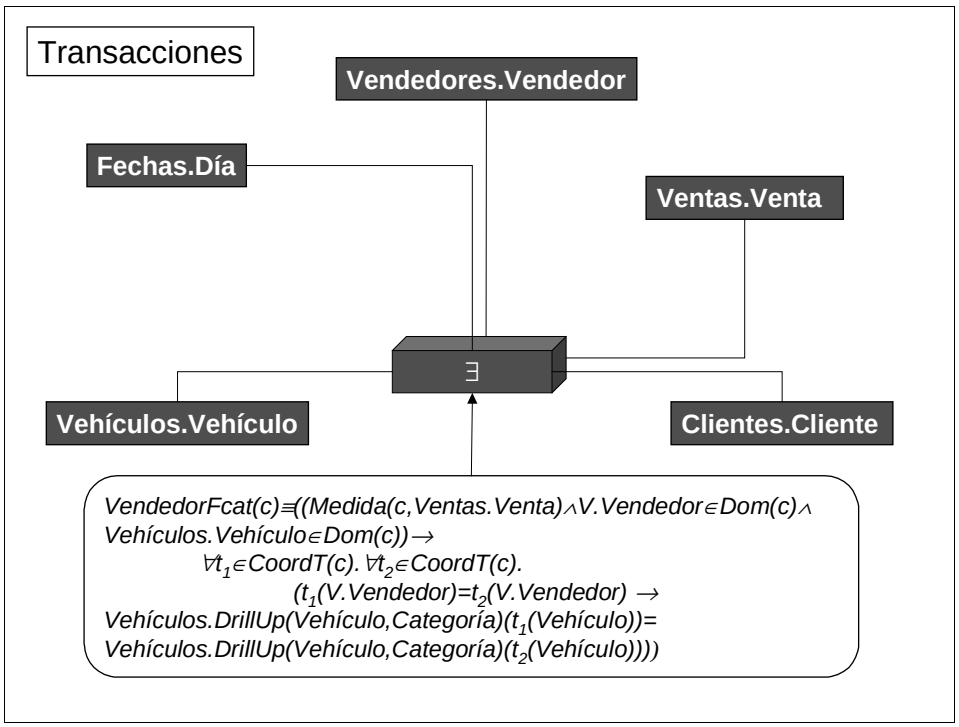

Fig. 29. Debe existir al menos un cubo en la instancia de Transacciones que cumpla con que los vendedores deben venden una sola categoría de vehículos. Ese cubo, tiene que tener en su dominio los niveles indicados en el diagrama.

El código del ejemplo de la restricción presentada anteriormente es el siguiente, sin considerar la declaración de la macro:

```
∃c.((Dom(c)= { Clientes.Cliente, Vehículos.Vehículo,
                Fechas.día, Vendedores.Vendedor,
                Ventas.Venta
                })∧
     VendedorFcat(c)
)
```

En caso de que la cuantificación fuera universal, la diferencia entre asociar la condición a un cubo o a una relación dimensional directamente, es que en el cubo se especifica el dominio.

Las restricciones generales, deben presentarse como macros con parámetros en donde el primero será el parámetro a cuantificar.

4.3.4.5. Especificación de Roll-up's.

Para especificar un roll-up es necesario indicar qué se debe calcular frente a qué drill-up. Para ello, se puede pensar que cuando se realiza un drill-up por determinada dimensión se produce un evento que alguien captura y como respuesta hace determinados calculos.

La notación propuesta para el drill-up se basa en ese tipo de visión (Fig. 30).

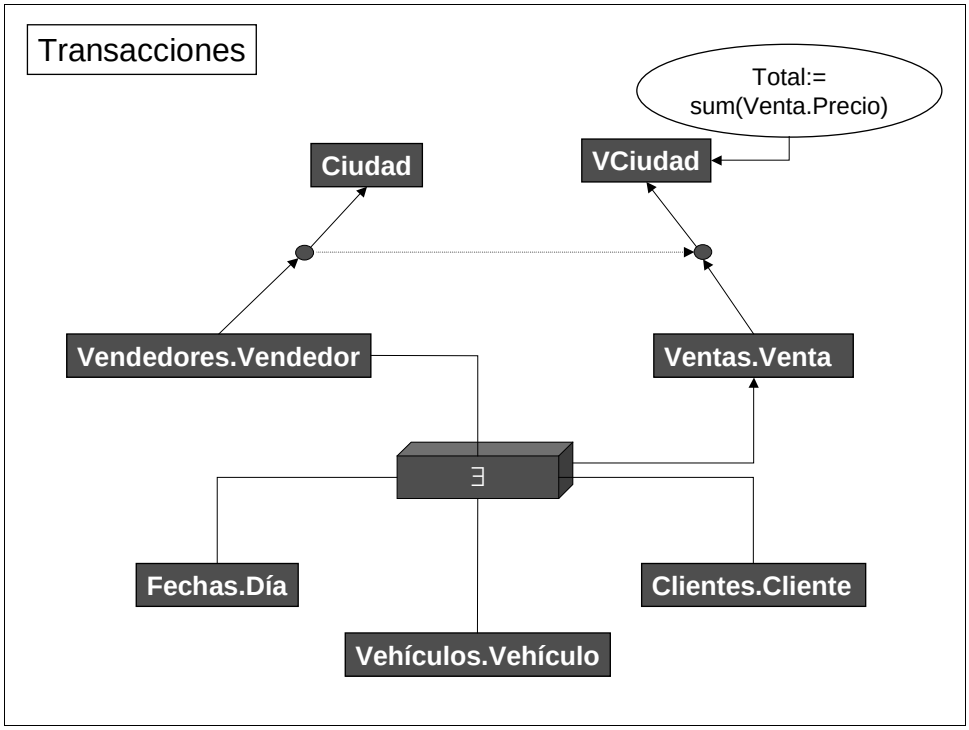

Fig. 30. Dado el cubo definido en el existencial, se sabe que existe otro que es el resultado de calcular la suma de los precios de venta de las unidades vendidas en cada ciudad

El código asociado para el caso de este ejemplo es el siguiente:

$$\exists c.(Cubo(\ c,\ \{Clientes.Cliente, Vehículos.Vehículo, Fechas.día, Vendedores.Vendedor\}, Ventas.Venta\ ) \wedge \exists c_1.IsRollUpOne(c, Vendedores, Vendedor, Ciudad, Ventas, Vciudad, Vcsum, c_1))$$

En donde *Vcsum* es la siguiente macro:

$$Vcsum(S) \equiv mk\_Vciudad(ValueOf(Total, sum(\{v/\exists o \in S.v=o.precio\})))$$

Otro mecanismo que puede ser útil es permitir especificar que cualquier drill-up por arriba de determinado nivel se pueda resolver con el mismo operador.

Para esto se usa la misma notación que en el caso anterior, sólo que se omite el nivel superior al que se desea llegar como resultado del drill-up (Fig. 31).

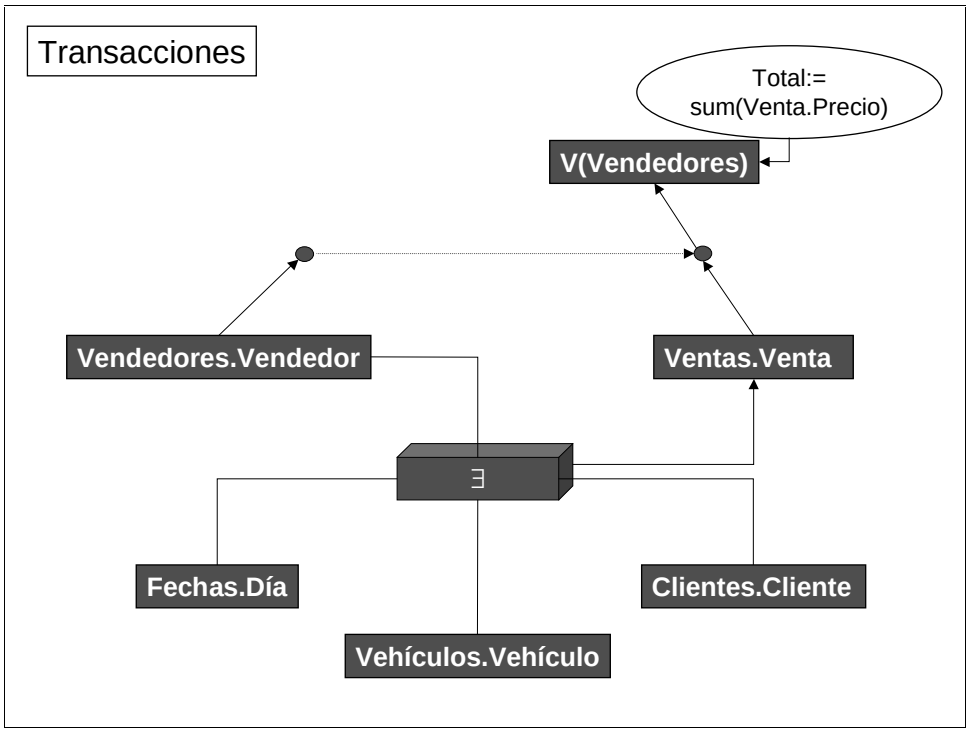

Fig. 31. Cualquier Drill-up por encima de Vendedor sonbre la dimensión Vendedores se resuelve con el operador ***sum****.*

La expresión asociada para este caso es:

$$
\begin{aligned}
&\exists c.(\ Cubo(\ c,\\
&\qquad \{\ Clientes.Cliente, Vehículos.Vehículo,\ Fechas.día,\\
&\qquad Vendedores.Vendedor\},\\
&\qquad Ventas.Venta\\
&\quad)\wedge\\
&\quad (\forall l \in Levels(Vendedores).(Vendedores.Op(Vendedor,l)\rightarrow\\
&\qquad \exists l_1 \in Levels(Ventas).(Ventas.Op(Venta,l_1)\wedge(l_1=\text{"V\_"}+l)\wedge\\
&\quad \exists c_1.IsRollUpOne(c,Vendedores,Vendedor,l,\\
&\qquad Ventas,Venta,l_1,Vcsum,c_1)\\
&)
\end{aligned}
$$

Notar que los cálculos siempre se hacen sobre el nivel *ventas*.

4.3.5. Conclusiones.

El lenguaje de restricciones, si bien es complejo a la hora de utilizarlo, permite especificar una semántica bastante precisa.

La estrategia de utilizar un lenguaje y macros sobre el mismo parece ser la adecuada en un marco exploratorio, es decir, en un marco en el cual aún no está claro que restricciones deben quedar como standard y cuáles no.

Además, la utilización de macros permite definir diferentes conjuntos de restricciones standard adecuados a diferentes realidades o simplemente para ayudar en la definición de restricciones tal como se manejaron en los ejemplos.

Si bien el agregado de notación gráfica para las restricciones puede resultar tedioso, el mismo debería facilitar la utilización del modelo.

Lo ideal, sería disponer de una herramienta que simplifique la edición del modelo incluyendo la edición del lenguaje de restricciones en forma gráfica. Dicha herramienta esta en construcción en este momento.

4.4. Formalización del Lenguaje de Restricciones.

En la primera sección, se estudian las características del lenguaje para fundamentar su definición. En las secciones restantes se define la sintaxis y semántica del lenguaje.

4.4.1. Análisis del Lenguaje.

El lenguaje utilizado en la sección anterior para expresar las restricciones tiene algunas características particulares que se analizan a continuación.

4.4.1.1. Lenguaje de Alto Orden.

El orden del lenguaje puede observarse en la siguiente macro:

$$Coord(c) \equiv \{f \;/\; \forall l \in Dom(c).\exists v \in l.f(l)=v\}$$

La variable *l* toma valores entre los niveles del dominio de un cubo *c* determinado. La variable *v* toma valores entre los elementos de la instancia del nivel *l*.

Esta característica indica que el lenguaje es de al menos, segundo orden.

El alto orden le da al lenguaje una potencia nada despreciable a la hora de especificar una realidad. Gracias a esta característica, es posible especificar condiciones utilizando conjuntos de cualquier tipo. Sin embargo, no es clara la utilidad de estas condiciones, de la misma forma que no es clara la posibilidad de prescindir de ellas.

Siguiendo el criterio planteado anteriormente, se prefirió dar una definición más general y, si es necesario, luego restringirla.

Por este motivo, se ha optado por definir un lenguaje de alto orden.

4.4.1.2. Términos Tipeados.

Las expresiones de la forma *f(f)* o $s \in s$ introducen problemas semánticos derivados de la paradoja de Russell. Para evitar estas expresiones, los términos deben tener tipo[6].

En la definición del lenguaje se considera un único tipo base, o sea, el tipo de los objetos más básicos que se manipulan con el lenguaje es único. Como se verá más adelante, esto no quita generalidad a la propuesta, pudiendo extenderse a más de un tipo básico.

4.4.1.3. Lenguaje de Restricciones o Lógica?

En esta sección se define un lenguaje para expresar restricciones de integridad. No se pretende definir un lenguaje lógico general y primitivo que soporte algún tipo de teoría específica, más allá de la necesaria para expresar las restricciones de integridad.

Por este motivo, para la definición de la semántica se toma como base la lógica clásica y la teoría de conjuntos.

4.4.1.4. Abreviaturas.

En los ejemplos que se manejaron anteriormente, existen un conjunto de abreviaturas como la omisión del conjunto en que toman valores las variables dependiendo del contexto y la utilización del símbolo "**.**" para indicar la selección tanto de atributos como la selección de componentes de la instancia como, por ejemplo, en *Vendedores.DrillUp*.

En esta sección se presenta la definición del lenguaje completo, es decir, sin abreviaturas.

4.4.1.5. Macros.

La definición no considera la existencia de las macros. Esto se debe a dos motivos:

- Las macros deberían procesarse antes de hacer cualquier otro procesamiento al lenguaje. Por lo tanto, no forman parte del mismo.
- El procesamiento de las macros no tiene ningún elemento particular. Debería poder realizarse, más allá de la sintaxis concreta de la definición y la aplicación de la macro, mediante cualquier procesador standard como el del lenguaje C o a lo sumo como m4.

Desde este punto de vista, el procesamiento de las macros debe considerarse como un problema resuelto a la hora de implementar algún procesamiento del lenguaje.

[6] No confundir estos "tipos" con los tipos de los niveles. En la formulación que se presenta del lenguaje, todos los objetos de los niveles son de un único tipo.

4.4.1.6. Familia de Lenguajes.

Por la forma en que se formaliza el lenguaje, en realidad se formaliza una familia de lenguajes. Existen conjuntos considerados elementales en la especificación, que si bien tienen algunos elementos fijos, el resto no siempre es el mismo.

La idea es que en el conjunto de las constantes estén todos los elementos del esquema, como por ejemplo, cada una de las dimensiones, niveles, relaciones dimensionales, etc. Estos elementos van a variar de esquema en esquema. Sin embargo, hay otras constantes que siempre deberán estar como por ejemplo las funciones de acceso al esquema (Levels, Dims, etc). Más adelante se discutirá este tema con mayor profundidad.

4.4.1.7. Restricciones y Macros.

No cualquier fórmula del lenguaje será una restricción válida. Sólo lo serán las fórmulas cerradas, es decir, sin variables libres.

Tampoco cualquier fórmula debería ser usada en una definición de macro. Sólo deberían ser variables libres en la fórmula los identificadores que sirven como parámetros de la macro.

4.4.2. Sintaxis Abstracta.

Por simplicidad de la presentación, la definición de la sintaxis del lenguaje se hace mediante conjuntos inductivos y no mediante BNF u otro mecanismo similar. Estas definiciones deben entenderse como una sintaxis abstracta, más allá de los detalles concretos que se presenten.

La definición se hace en tres secciones.

- Primero se define el lenguaje de los tipos de los términos.
- Luego se define el lenguaje de los términos en sí mismos, indicando al mismo tiempo las reglas de tipeado de los mismos.
- Por último se definen las expresiones del lenguaje.

4.4.2.1. Los Tipos de los Términos.

En el lenguaje, existen cuatro clases de términos:

- Los que representan elementos de las instancias de los niveles o elementos de tipos básicos como números o strings. Estos términos tendrán tipo **i.**
- Los que representan funciones de acceso o de algún otro tipo. El tipo de estos términos será funcional y por lo tanto será una expresión basada en el constructor →.
- Los que representan conjuntos. La expresión del tipo de estos términos estará basada en el constructor **Set**.
- Los que representan tuplas[7]. Estarán basados en el constructor **<,>**.

[7] Ver la macro *IsRollUpSim* en la sección 4.3.3.3

El lenguaje de las expresiones de tipo de términos se puede definir como en la Fig. 32.

I.	$\underline{i} \in TermTypes;$
II.	$\alpha_i \rightarrow \alpha_j \in TermTypes$ si $\alpha_i, \alpha_j \in TermTypes;$
III.	$Set(\alpha) \in TermTypes;$
IV.	$\underline{[}\alpha_o, ..., \alpha_n] \in TermTypes$ si $\alpha_o, ..., \alpha_n \in$

Fig. 32. Definición de los Tipos de los Términos.

4.4.2.2. Los Términos.

Para definir los términos, se asume la existencia de dos familias de conjuntos indizadas por expresiones de tipo:

- **Const** es la familia de los conjuntos de constantes de un tipo dado. Dicho de otra forma, **Const(α)** es el conjunto de las constantes de tipo α.
- **Var** es la familia de los conjuntos de variables de un tipo dado. Dicho de otra forma, **Var(α)** es el conjunto de las variables de tipo α.

Con estas suposiciones previas, el conjunto de los términos se define en la Fig. 33.

Las cláusulas I y II definen las constantes y las variables de cualquier tipo respectivamente.

La cláusula III define las tuplas de términos como un nuevo término y la cláusula IV define la aplicación de un término a otro.

I. $C:\alpha \in$ Term si $C:\alpha \in$ CONST(α);

II. $X:\alpha \in$ Term si $X:\alpha \in$ VARS(α);

III. $<t_1,\ldots,t_n>:[\alpha_1,\ldots,\alpha_n] \in$ Term si $t_1:\alpha_1,\ldots,t_n:\alpha_n \in$ Term;

IV. $t(t_1):\alpha \in$ Term si $t:\alpha_1 \rightarrow \alpha \in$ Term y $t_1: \alpha \in$ Term;

V. $\{c_1 \ldots c_n\}:Set(\alpha) \in$ Term si $c_1:\alpha \in$ CONST,…,$c_n:\alpha \in$ CONST;

VI. $\{ t(x_1,\ldots,x_j) / \varphi(x_1,\ldots,x_n)\}:Set(\alpha) \in$ Term si $t:\alpha \in$ Term, S:Set(α) ∈ Term, $\varphi \in$ Form y FV(t) ⊆ FV(φ)donde FV es el conjunto de las variables libres que aparecen en un término o en una fórmula (j<i).

Fig. 33. Definición de los Términos con su tipo.

Las cláusulas V y VI permiten la definición de conjuntos por extensión y por comprensión respectivamente. Observar que cuando se definen conjuntos por comprensión, no hay variables libres en el término no sean también libres en la fórmula.

Es importante notar que la definición de los términos, es mutuamente recursiva con las fórmulas lógicas debido a la posibilidad de definir conjuntos por comprensión.

La definición de la función FV se puede ver en Fig. 34.

I. $FV(c)=\{\}$

II. $FV(x)=\{x\}$

III. $FV(<t_1,\ldots,t_n>)=\cup_{l=1,n}FV(t_i)$

IV. $FV(t(t_1))=FV(t)\cup FV(t_1)$

V. $FV(\{t_1,\ldots,t_n\})=\cup_{l=1,n}FV(t_i)$

VI. $FV(\{t/\varphi\})=FV(\varphi)-FV(t)$

Fig. 34. Definición de la función FV para los términos.

4.4.2.3. Las Fórmulas del Lenguaje.

Las fórmulas base que se presentan son de dos tipos:

- Predicados de pertenencia a conjuntos.
- Comparación de términos del mismo tipo.

La definición del lenguaje de las fórmulas basadas en estos predicados base se puede ver en Fig. 35.

I. $(\gamma \in \Gamma) \in$ Form, si $\gamma:\alpha \in$ Term y $\Gamma:Set(\alpha) \in$ Term;

II. $(\gamma_1 \oplus \gamma_2) \in$ Form, si $\gamma_1:\alpha$, $\gamma_2:\alpha \in$ Term y $\oplus \in$ BINCOMP;

III. $(\varphi_1 \oplus \varphi_2) \in$ Form, si $\oplus \in$ BINCONN y φ_1 , $\varphi_2 \in$ Form;

IV. $(\neg\varphi) \in$ Form, si $\varphi \in$ Form;

V. $(\forall x \in S.\varphi) \in$ Form, si $x:\alpha \in$ VARS(α), $S:Set(\alpha) \in$ Term y $\varphi \in$ Form;

VI. $(\exists x \in S.\varphi) \in$ Form, si $x:\alpha \in$ VARS(α), $S:Set(\alpha) \in$ Term y $\varphi \in$ Form;

Fig. 35. Definición del lenguaje de las Fórmulas para el Lenguaje de Restricciones.

En la definición quedan dos conjuntos sin definir. El conjunto BINCOMP es el conjunto de los símbolos de comparaciones binarias. El conjunto BINNCONN es el conjunto de los conectores lógicos binarios. Mientras que BINNCONN es fijo y contiene los símbolos $\wedge$, $\vee$, $\rightarrow$ y $\leftrightarrow$, el conjunto BINCOMP puede ser considerado como un parámetro en la definición. Se puede asumir que en principio tiene los símbolos =,>,<,≥,≤[8].

Al definir el conjunto de las fórmulas de esta forma, no es necesario que las mismas tengan tipo. Sin embargo, la definición fuerza el correcto tipado de los términos.

En la definición de los términos, se utiliza la siguiente función que calcula el conjunto de las variables libres de una fórmula (Fig. 36).

I. $FV((\gamma \in \Gamma))=FV(\gamma)\cup FV(\Gamma)$

II. $FV((\gamma_1 \oplus \gamma_2))=FV(\gamma_1)\cup FV(\gamma_2)$

III. $FV((\varphi_1 \oplus \varphi_2))=FV(\varphi_1)\cup FV(\varphi_2)$

IV. $FV((\neg\varphi))=FV(\varphi)$

V. $FV((\forall x\in S.\varphi))=FV(\varphi)-\{x\}$

VI. $FV((\exists x\in S.\varphi))=FV(\varphi)-\{x\}$

Fig. 36. Definición de la función FV para las Fórmulas.

4.4.3. Semántica.

Como se remarcó anteriormente, sólo se pretende definir un lenguaje potente para la expresión de restricciones de integridad y no un lenguaje lógico general.

A pesar de esto, se presenta la semántica del lenguaje en una forma similar a la que se utiliza para presentar la semántica de la lógica clásica. Si bien se podría haber utilizado otro mecanismo de especificación como podría ser semántica denotacional u operacional, se prefirió esta línea de especificación por su simplicidad (de formulación y comprensión) y su nivel de abstracción.

4.4.3.1. Semántica de los Términos.

La semántica de los términos se define en función un conjunto dependiente del tipo de los términos. Para esto, se define la función Dom_{Δ} que va del lenguaje de los tipos en conjuntos construidos sobre un conjunto Δ finito (Fig. 37). La idea básica es que los elementos de Δ son de tipo *i*, los subconjuntos de Δ son de tipo *set(i)* , los elementos de $\Delta X\Delta$ son de tipo *[i,i]*, los subconjuntos de $\Delta X\Delta$ son de tipo *set([i,i])*, etc.

[8] En realidad, debería ser una familia de conjun dependiente del tipo. De esa forma se estaría previendo la existencia de varios tipos base en los término

I. $Dom_{\Delta}(i)=\Delta$

II. $Dom_{\Delta}([\alpha_0,\ldots,\alpha_n])=(\{i / i\in N \wedge 0<=i<=n\}\rightarrow Dom_{\Delta}(\alpha_i))$

III. $Dom_{\Delta}(set(\alpha))=Pot(Dom_{\Delta}(\alpha))$

IV. $Dom_{\Delta}(\alpha_i\rightarrow\alpha_j)=(Dom_{\Delta}(\alpha_i)\rightarrow Dom_{\Delta}(\alpha_j))$

Fig. 37. Definición de la función Dom, que va del lenguaje de los tipos de los términos en conjuntos construidos sobre un conjunto inicial Δ.

Lo primero que se observa, es que si Δ es finito, entonces $Dom_{\Delta}(\alpha)$ es finito para α cualquiera.

La semántica para los términos, se define mediante una función V_{Δ} llamada función de interpretación o simplemente interpretación que mapea cada término en un elemento de ese universo tal que ese elemento es del tipo correcto.

Se asume la existencia de una familia de conjuntos indizada en los tipos de los términos $C(\alpha)$ tal que cada conjunto contiene elementos distinguidos de $Dom_{\Delta}(\alpha)$. También se asume la manipulación de lenguaje extendido, esto es que además de las constantes de la familia C, si $a\in Dom_{\Delta}(\alpha)$, entonces $\underline{a}$ es un término de tipo α.

La definición de la interpretación se puede ver en Fig. 38.

V_{Δ}:{ $t:\alpha$ / $t:\alpha \in Term\}\rightarrow Dom_{\Delta}(\alpha)$

I. $V_{\Delta}(c)=a$ donde $a \in C(\alpha)$ y c es una constante con tipo α.

II. $V_{\Delta}(\underline{a})=a$ donde $a \in Dom_{\Delta}(\alpha)$

III. $V_{\Delta}(<t_1,\ldots,t_n>)=f$ donde $t_1:\alpha_1,\ldots,t_n:\alpha_n$ y

$f \in Dom_{\Delta}(<\alpha_1,\ldots,\alpha_n>)$ y

$f(i)=V_{\Delta}(t_i)$ $\forall i.1<=i<=n$

IV. $V_{\Delta}(t(t_1)) = V_{\Delta}(t)(V_{\Delta}(t_1))$.

V. $V_{\Delta}(\{t_1\ldots t_n\}) =\{V_{\Delta}(t_1),\ldots,V_{\Delta}(t_n)\}$

VI. $V_{\Delta}(\{t / \varphi(t)\}) = \{V_{\Delta}(t[\underline{a_1}/x_1,\ldots,\underline{a_n}/x_n])/ (\forall i\in[1,n].a_i\in Dom(\alpha)) \wedge F_{\Delta}(\varphi(\underline{a})) =1\}$ donde $x:\alpha$.

Fig. 38. Definición de la función de Interpretación.

Hay dos puntos relevantes a recalcar sobre esta definición:

1. El producto cartesiano se ve como funciones de un rango de los naturales en los dominios correctos.
2. En la última cláusula se utiliza la función que determina la semántica de una fórmula del lenguaje.

Con esta visión de los términos, una estructura para interpretar el lenguaje que se está manejando, sólo necesita dos elementos: Δ y la familia C indizada en los tipos de los términos.

4.4.3.2. Semántica de las Fórmulas.

Para definir la semántica de las fórmulas, se pueden tomar dos enfoques levemente diferentes pero equivalentes.

Un enfoque es definir una función que dada una fórmula devuelve 0 si la fórmula no se satisface o 1 si la fórmula se satisface.

El otro enfoque es definir una función que dada una fórmula construye el conjunto de las interpretaciones restringidas[9] que las satisface.

En esa sección se sigue el primer enfoque.

F_Δ: Form $\rightarrow \{0,1\}$

$F_\Delta((t \in S))=1$ si $V_\Delta(t) \in V_\Delta(S)$, devuelve 0 en otro caso.

$F_\Delta((t_1 \oplus t_2))=1$ si la pareja $<V_\Delta(t_1),V_\Delta(t_2)>$ pertenece a la relación denotada por $\oplus$.

$F_\Delta((\varphi_1 \wedge \varphi_2))=\min(\{F_\Delta(\varphi_1),F_\Delta(\varphi_1)\})$

$F_\Delta((\varphi_1 \vee \varphi_2))=\max(\{F_\Delta(\varphi_1),F_\Delta(\varphi_1)\})$

$F_\Delta((\varphi_1 \rightarrow \varphi_2))=\max(\{1-F_\Delta(\varphi_1),F_\Delta(\varphi_1)\})$

$F_\Delta((\varphi_1 \leftrightarrow \varphi_2))=1-|F_\Delta(\varphi_1)-F_\Delta(\varphi_1)|$

$F_\Delta((\neg\varphi_1))=1-F_\Delta(\varphi_1)$

$F_\Delta((\forall x \in S.\varphi_1))=1$ si $V_\Delta(S)=\{\}$,

$\min(\{F_\Delta(\varphi_1[\underline{a}/x])/a \in V_\Delta(S)\})$ si $V_\Delta(S)\neq\{\}$.

$F_\Delta((\exists x \in S.\varphi_1))=0$ si $V_\Delta(S)=\{\}$,

$\max(\{F_\Delta(\varphi_1[\underline{a}/x])/a \in V_\Delta(S)\})$ si $V_\Delta(S)\neq\{\}$.

Fig. 39. Definición de la Intepretación de las Fórmulas del Lenguaje de Restricciones.

4.4.4. Otros Elementos del Lenguaje de Restricciones.

Hay ciertos elementos que aún no se han discutido lo suficiente.

Si bien se mencionó anteriormente, no se ha presentado la relación entre las restricciones y el esquema.

Tampoco se ha discutido, que son en este contexto las funciones e agregación.

Otro elemento pendiente es cómo extender el lenguaje para soportar varios tipos de datos básicos.

En esta sección se discuten estos puntos.

[9] El término "restringidas" se refiere a la restricción del dominio de la función a los términos que aparecen en la fórmula.

4.4.4.1. Restricciones, Esquemas e Instancias.

La relación entre el esquema y las restricciones está dada por dos elementos básicos:

- El esquema determina el contexto de una restricción de forma de poder resolver las abreviaturas que se estén manejando.
- Las constantes que pueden aparecer en las restricciones son o bien funciones de acceso al esquema, o bien elementos del esquema.

Las constantes que se refieren a elementos del esquema, se deben interpretar como la instancia de ese elemento. De esta forma cada constante tiene un tipo coherente con la instancia del objeto que representan.

Los niveles representan conjuntos de elementos de Δ, por lo que tienen tipo ***Set(i)***.

Las dimensiones representan parejas formadas por una función que mapea los nombres de los niveles en instancias del nivel y una función que dados dos niveles devuelve otra función que si se le da un elemento de un nivel, devuelve un elemento del nivel superior. El tipo de las dimensiones, entonces debe ser ***[Set(i)→Set(Set(i)),[Set(i),Set(i)]→(i→i)]***.

Las relaciones dimensionales representan un subconjunto de los cubos que se pueden formar tomando al menos un nivel de cada una de las dimensiones.

Para determinar el tipo de las relaciones dimensionales, vale la pena analizar con cuidado la especificación de sus instancias. La instancia de una relación dimensional es un conjunto de funciones. Esas funciones van de las coordenadas posibles en los booleanos. Por esto, el tipo de las relaciones dimensionales debería ser:

Set((Set(i)→i)→i)

Una constante introducida por el esquema, siempre representa (se interpreta como) la instancia de ese elemento del esquema.

Además de las constantes que representan elementos del esquema, se necesitan constantes que representan funciones para acceder a diversos elementos del esquema y las instancias. Por ejemplo, la siguiente función que devuelve la función de DrillUp de una dimensión:

DrillUp*:*([Set(i)→Set(i),[Set(i),Set(i)]→(i→i)])→([Set(i),Set(i)]→(i→i)))

En el anexo II, se presenta el conjunto de constantes que se deben introducir Para cada esquema.

4.4.4.2. Funciones de Agregación.

Típicamente, una función de agregación es una función que recibe un conjunto y devuelve un elemento de algún otro conjunto.

Las más tipicas son Sum y Avg. La primera devuelve, dado un conjunto de números devuelve su suma y la segunda devuelve su promedio. En términos de los tipos de los términos, si existieran estas constantes deberían tener todas el mismo tipo: ***Set(i)→i***.

En el contexto del lenguaje de restricciones, la noción de función de agregación es un poco más general: son funciones con dominio ***Set(i)*** .

En el anexo II se presenta una visión un poco más precisa de las funciones de agregación.

4.4.4.3. Soporte de Múltiples Tipos de Datos.

En todo lo que se ha presentado del lenguaje de restricciones, se parte de la base de la existencia de un único tipo de datos *i* cuyos elementos están en Δ y al cual pertenecen todos los elementos básicos necesarios.

Hay al menos dos formas de incluir el soporte de varios tipos de datos:

- Considerar Δ como una familia de conjuntos indizada por la identificación de los tipos básicos. Esto afectaría la definición de la función *Dom*.
- Considerar que cada tipo no es más que un subconjunto específico de Δ. Esto es equivalente a pensar en Δ como el conjunto de los identificadores de objeto en un lenguaje orientado a objeto, y suponer la existencia de una función que dado el objeto puede devolver el tipo del mismo. De la misma forma, las operaciones que se aplican al objeto dependerán del tipo (en este caso, la clase) del objeto.

En el contexto de la propuesta parece adecuado seguir el segundo enfoque. Esto lleva a que las funciones específicas de cada tipo se introduzcan al lenguaje como nuevas constantes de tipo $\boldsymbol{i}\rightarrow\alpha$ que funcionarían como las selectoras de los nuevos tipos. Además, se asume que para cada nivel existe una función tal que dados determinados parámetros construye un elemento del nivel.

De esta forma, el lenguaje sigue soportando un único tipo básico para los términos pero soporta múltiples tipos de datos para los niveles.

En los ejemplos presentados en las secciones anteriores se asume que los niveles tienen tipo producto cartesiano con nombres en componentes (tipo *tupla*), soportando básicamente tipos numéricos y string.

4.5. Formalización de CMDM con Restricciones.

Las restricciones de integridad son condiciones que se imponen sobre las estructuras de datos que conforman el modelo. Por esto cada estructura de datos debe tener asociado un conjunto de restricciones.

4.5.1. Declaraciones.

Una declaración, además de asociar un tipo con un nombre de nivel, debe asociar un conjunto de restricciones.

Definición 12. Declaraciones de Nivel con Restricciones.

L es un conjunto de declaraciones de esquemas de niveles con restricciones si pertenece al siguiente conjunto:

Declarations ≡ **{ <Level_Names,Type,Consts> /**
Level_Names ∈ **Set(Strings)** ∧
Type ∈ **Level_Names** → **ETYPE** ∧
Consts ∈ **Level_Names**→**Set(Form)**
}

Además, las instancias de cada nivel deben cumplir las restricciones:

Definición 13. Instancia de Nivel con Restricciones.

Si **E** es un Esquema Multidimensional y **l** es un nivel con restricciones, entonces **Inst(l)** es una instancia de **l** si pertenece al siguiente conjunto:

Level_SInst(E,l) ≡ **{c / E** ∈ **DimSch** ∧ **l** ∈ **E.Decls.Level_names** ∧
c∈ **Set(EType2Set(E.Decls.Type(l)))** ∧ **Check(E.Decls.Consts(l))**
}

Donde **Check** es un predicado que devuelve verdadero si son verdaderas cada una de las restricciones del conjunto que recibe como parámetro.

4.5.2. Dimensiones.

Al igual que los niveles, una dimensión tiene asociado un conjunto de restricciones que deben ser cumplidas por sus instancias.

Definición 14. Esquema de Dimensión con Restricciones.

D es un *Esquema de Dimensión con Restricciones* sobre las declaraciones **Decls**, si pertenece al siguiente conjunto

Dimensions(Decls) ≡ **{ <DimName, L, Po,Consts> /**
DimName ∈ **Strings** ∧ **L** ∈ **Set(Decls.Level_Names)** ∧
Po∈ **Pos (L)** ∧ **Consts**∈ **Set(Form)}**

Definición 15. Instancia de una Dimensión con Restricciones.

Si **E** es un esquema multidimensional y **D** es un esquema de dimensión con restricciones, entonces **inst(D)** es una instancia de **D** si pertenece al siguiente conjunto:

$$\begin{aligned}&\textbf{Dim\_SInst(E,D)} \equiv \{ \langle il,du\rangle \;/\; E\in \textbf{DimSch} \wedge D\in \textbf{Dimensions(E.Dims)} \wedge \\ &\quad \forall(l\in \textbf{D.L}).(il(l) \in \textbf{Level\_SInst(E,l)}) \wedge \\ &\quad (du \in \{\langle l_1,l_2\rangle \in \textbf{D.Po}\} \rightarrow (il(l_1)\rightarrow il(l_2)))\wedge \\ &\quad \forall(l\in \textbf{D.L}).\forall(l_1\in \textbf{D.L}).\forall(l_2\in \textbf{D.L}). \\ &\quad ((\langle l,l_1\rangle\in \textbf{D.Po}\wedge\langle l_1,l_2\rangle\in \textbf{D.Po})\rightarrow \\ &\forall o\in l.(du(l_1,l_2)(du(l,l_1)(o))=du(l,l_2)(o))) \wedge \textbf{Check(D.Consts)} \}\end{aligned}$$

4.5.3. Relaciones Dimensionales.

También las relaciones dimensionales necesitan un conjunto de restricciones que las instancias deben satisfacer.

R es un *Esquema de Relación Dimensional con Restricciones*, si pertenece al siguiente conjunto:

$$\begin{aligned}&\textbf{Relations(Dims)} \equiv \{\langle \textbf{RelName,D,Consts}\rangle \;/\; \textbf{RelName}\in \textbf{Strings} \wedge \\ &\quad \textbf{D} \in \textbf{Set(Dims)} \wedge \textbf{Consts}\in \textbf{Set(Form)}\}\end{aligned}$$

Definición 16. Esquema de Relación Dimensional con Restricciones.

Definición 17. Instancia de una Relación Dimensional con Restricciones.

Si **E** es un esquema multidimensional y **RD** es un esquema de relación dimensional, entonces **inst(RD)** es una instancia de **RD** si pertenece al siguiente conjunto:

$$\begin{aligned}&\textit{RD\_Sinst(E,RD)} \equiv \{ \textit{sf} \;/\; \textit{E} \in \textit{DimSch} \wedge \textit{RD} \in \textit{RelDims} \wedge \\ &\quad \exists(\textit{L}\in \textit{RD\_LevelSet(E,RD)}).(\textit{sf} \in \textit{Set(Coord(E,L)} \rightarrow \textit{Boolean}))\wedge \\ &\quad \textit{Check(RD.Consts)}\}\end{aligned}$$

4.5.4. Esquema Multidimensional.

Una restricción que involucre dos relaciones dimensionales, no debería ser ubicada con ninguna de esos ellas. Por esto, los esquemas también tienen asociado un conjunto de restricciones.

Definición 18. Esquema Multidimensional con Restricciones.

E es un *Esquema Multidimensional* si pertenece al siguiente conjunto:

$$\begin{aligned}\textbf{DimSch} \equiv \{\ &\textbf{<Name, Decls, Dims, Rels, Consts > / Name} \in \textbf{Strings} \wedge \\ &\textbf{Decls} \in \textbf{Declarations} \wedge \\ &\textbf{Dims} \in \textbf{Set(Dimensions(Decls))} \wedge \\ &\textbf{Rels} \in \textbf{Set(Relations(Dims))} \wedge \textbf{Consts} \in \textbf{Set(Form)} \\ \}\end{aligned}$$

La instancia de un esquema multidimensional, debe cumplir con sus restricciones.

Definición 19. Instancia de un Esquema Multidimensional con Restricciones.

Si **E** es un esquema multidimensional, entonces **inst(E)** es una instancia de ese esquema si pertenece al siguiente conjunto:

$$\begin{aligned}&\textbf{Sinst\_Sch(E)} \equiv \{\textbf{<IL,ID,IR,Consts> /} \\ &\quad \textbf{IL} \in \{\textbf{l/l} \in \textbf{E.Decls.Level\_names}\} \rightarrow \textbf{Sinstance(E,l)} \wedge \\ &\quad \textbf{ID} \in \{\textbf{d/d} \in \textbf{E.Dims}\} \rightarrow \textbf{Sinstance(E,d)} \wedge \\ &\quad \textbf{IR} \in \{\textbf{r/r} \in \textbf{E.RelDim}\} \rightarrow \textbf{Sinstance(E,r)} \wedge \textbf{Check(Consts)} \\ \}\end{aligned}$$

4.6. Algunas Conclusiones Preliminares.

El lenguaje de restricciones permite la especificación de una base multidimensional con un nivel de detalle similar al que se puede obtener al realizar una especificación en Modelo Entidad Relación con restricciones no estructurales.

No hay en el modelo un mecanismo para especificar la relación entre las estructuras multidimensionales y las estructuras de las bases fuentes.

El modelo no se pliega a ninguna metodología específica. Parece ser viable tanto para una forma de trabajo tradicional, en el sentido de construir el diseño del DW o Datamarts a partir de un conjunto de bases fuentes. También parece ser viable para realizar una forma de trabajo inversa a la tradicional: construir una especificación del DW o de los Datamarts y de allí obtener cuáles son las bases fuentes que se deben construir.

El lenguaje gráfico presentado incluyendo las representaciones gráficas de las restricciones de integridad, debe ser considerado como un prototipo de lenguaje. Es decir, las estructuras y las restricciones de integridad son lo sustancial del modelo y el lenguaje gráfico se puede ir cambiando o adaptando con el tiempo a una representación mejor que la actual, más allá de que no se introduzcan demasiados cambios en las estructuras ni en el lenguaje de restricciones.

Por otro lado, parece imprescindible para su utilización real, la existencia de una herramienta CASE que permita la edición y el procesamiento del modelo.

En la siguiente sección se presenta una descripción de una herramienta CASE, de la cual, ya hay algunas partes construidas.

Capítulo 5. Prototipo.

5.1. Introducción.

En esta sección se presenta la arquitectura de un ambiente CASE basada en CMDM y orientada al diseño de aplicaciones Olap y de Datawarehousing.

Esta herramienta está siendo desarrollada por el Laboratorio de Concepción de Sistemas de Información. Este laboratorio enmarca su trabajo dentro del Instituto de Computación de la Facultad de Ingeniería, dependiente de la Universidad de la República Oriental del Uruguay.

El objetivo fundamental de la herramienta es brindar apoyo al analista en el desarrollo de sistemas basados en Data Warehouse. Para ello la herramienta trata de cubrir la mayor parte de los elementos que participan en un sistema de Data Warehousing brindando algún servicio específico para auxiliar al analista en el desarrollo de ese elemento en concreto. El analista puede utilizar la herramienta siguiendo tanto una metodología top-down como bottom-up.

En este sentido, se puede decir que el ambiente está orientado al tipo de sistema más que a la metodología.

5.2. Arquitectura del Ambiente CASE.

La arquitectura del ambiente se basa en un conjunto de diferentes módulos diseñados y programados en forma totalmente independiente y luego integrados a través de una interfase CORBA.

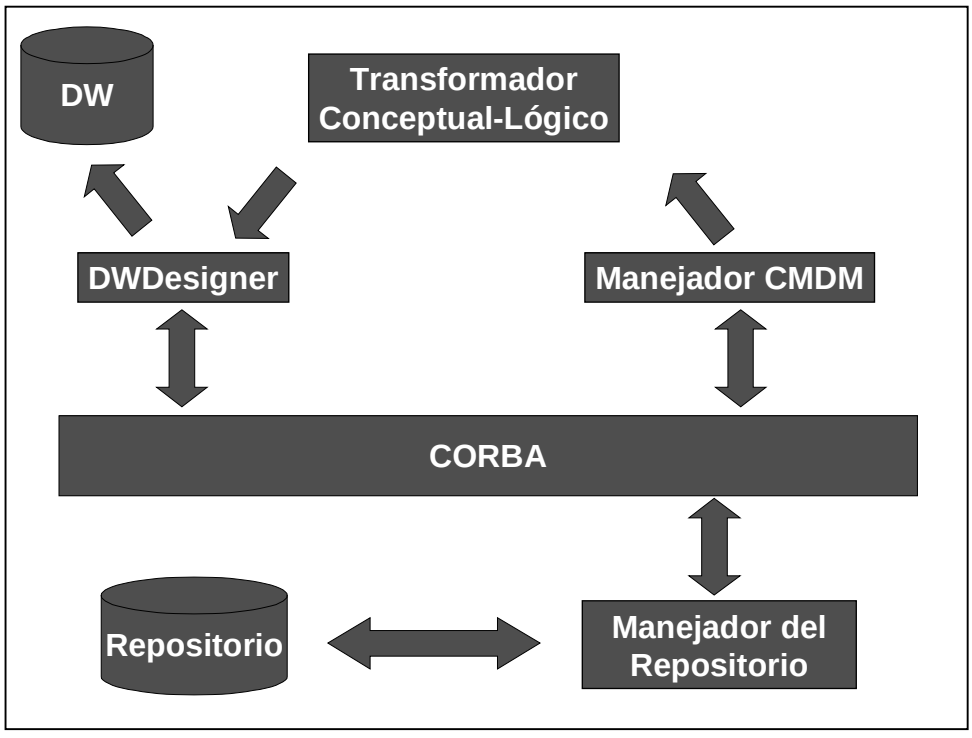

Fig. 40. Arquitectura de una Herramienta CASE orientada al Diseño de Datawarehouses.

Durante el transcurso de este trabajo se diseñó el módulo ***Manejador CMDM***. En las siguientes secciones se describen la arquitectura, el diseño y la implementación de un prototipo de dicho módulo.

5.3. Manejador Cmdm.

Este módulo se encarga del manejo del modelo conceptual. Actualmente se dispone de un prototipo parcial de este módulo, construido por estudiantes en su proyecto de grado.

Su arquitectura se puede ver en la figura Fig. 41

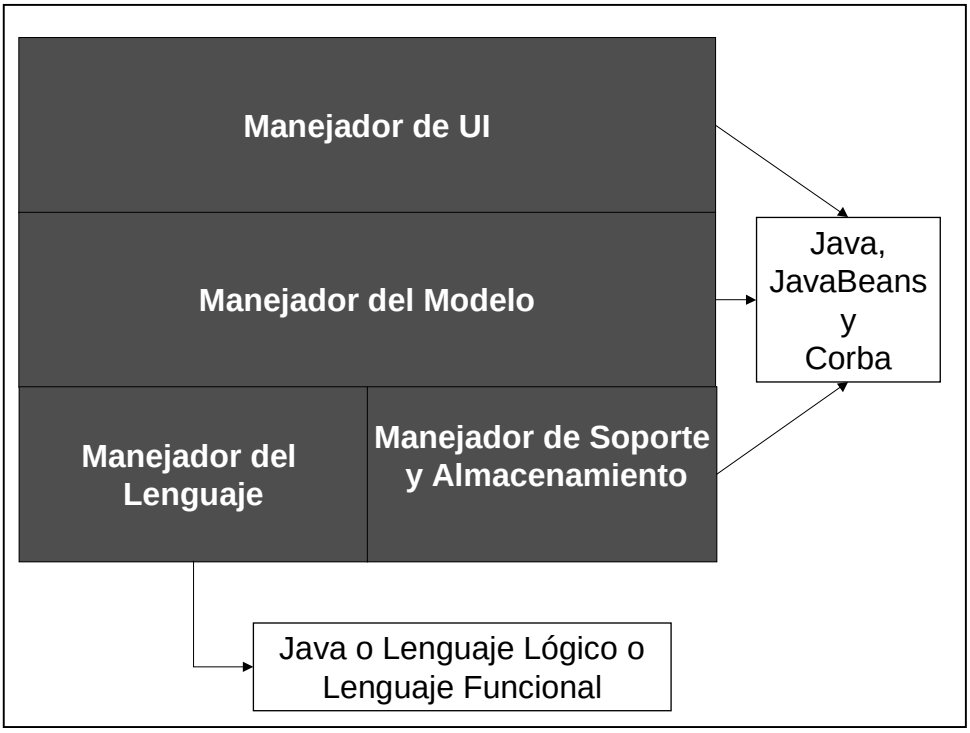

Fig. 41. Arquitectura del Manejador CMDM.

El módulo permite construir una especificación en CMDM y provee interfases con otros módulos que permitan el procesamiento de dicha especificación. De esta forma, el usuario posee una interfase gráfica desde la cual puede acceder a todos los servicios necesarios para editar los diagramas y agregar restricciones. También se proveen servicios orientados a obtener información del modelo creado y generar código para diferentes aplicaciones.

El usuario trabaja siempre sobre una interfase que es proporcionada por el módulo *Manejador de IU*. Este módulo es una interfase gráfica genérica programada en Java y capaz de soportar JavaBeans.

El módulo *Manejador del Modelo* provee los Beans que permiten construir los diagramas en el *Manjeador de IU*. Cada Bean es un objeto que representa algún elemento del modelo, como por ejemplo, un nivel o una dimensión. Cada uno de estos Beans, provee información sobre como debe presentarse y aporta su parte al comportamiento del editor. La ventaja de esta arquitectura es que para cambiar algo en el modelo, sólo basta tocar el *Manejador del Modelo* sin que esto obligue a cambiar el *Manejador de IU* aunque se cambien elementos de presentación de alguna parte del modelo.

El módulo *Manejador de Soporte y Almacenamiento* provee funciones básicas para guardar el modelo en disco local, comunicarse con el repositorio para recuperar o guardar un modelo y también funciones básicas para la generación de reportes, impresión de los diagramas, etc. La conexión con el repositorio se realiza mediante CORBA.

El módulo *Manejador del Lenguaje* se encarga de realizar chequeos sintácticos y semánticos sobre el esquema y, además, provee la generación de código para diferentes aplicaciones multidimensionales o relacionales.

5.3.1. Prototipo Actual.

En Fig. 42 se muestra la apariencia del prototipo.

El *Manejador de IU* está implementado casi en su totalidad. Este módulo provee las funciones típicas de un editor como cortar, copiar, pegar,etc. En particular, su ventana principal se divide en dos partes: un visor jerárquico y un área de trabajo. El visor jerárquico provee una forma rápida de navegación entre todos los elementos del modelo. En el área de trabajo se realiza la edición de cada componente del modelo mediante la utilización de los Beans que son provistos por el *Manejador del Modelo*.

También se posee una primera versión del *Manejador del Modelo* en donde los tipos de los niveles y las restricciones de cualquier elemento del modelo son strings. Por otro lado, provee algunas extensiones al modelo como las “jerarquías macro” y condición de “estructura pendiente”. El *Manejador del Modelo* tiene la posibilidad de crear la *Visión por Relaciones Dimensionales*, en donde se calculan los caminos de Drill-Across a través de las dimensiones.

El *Manejador de Soporte y Almacenamiento* provee funciones para trabajar en disco local pero aún no provee la comunicación con el repositorio. A estas funciones se puede acceder desde los menúes provistos por el *Manejador de UI*.

Una primera versión del *Manejador del Lenguaje* se está implementando en Gnu Prolog. La elección de un lenguaje como Prolog para este módulo se fundamenta en la necesidad de mecanismos de programación que permitan la implementación rápida de mecanismos de procesamiento sintáctico y semántico del lenguaje de restricciones y de la definición de las estructuras.

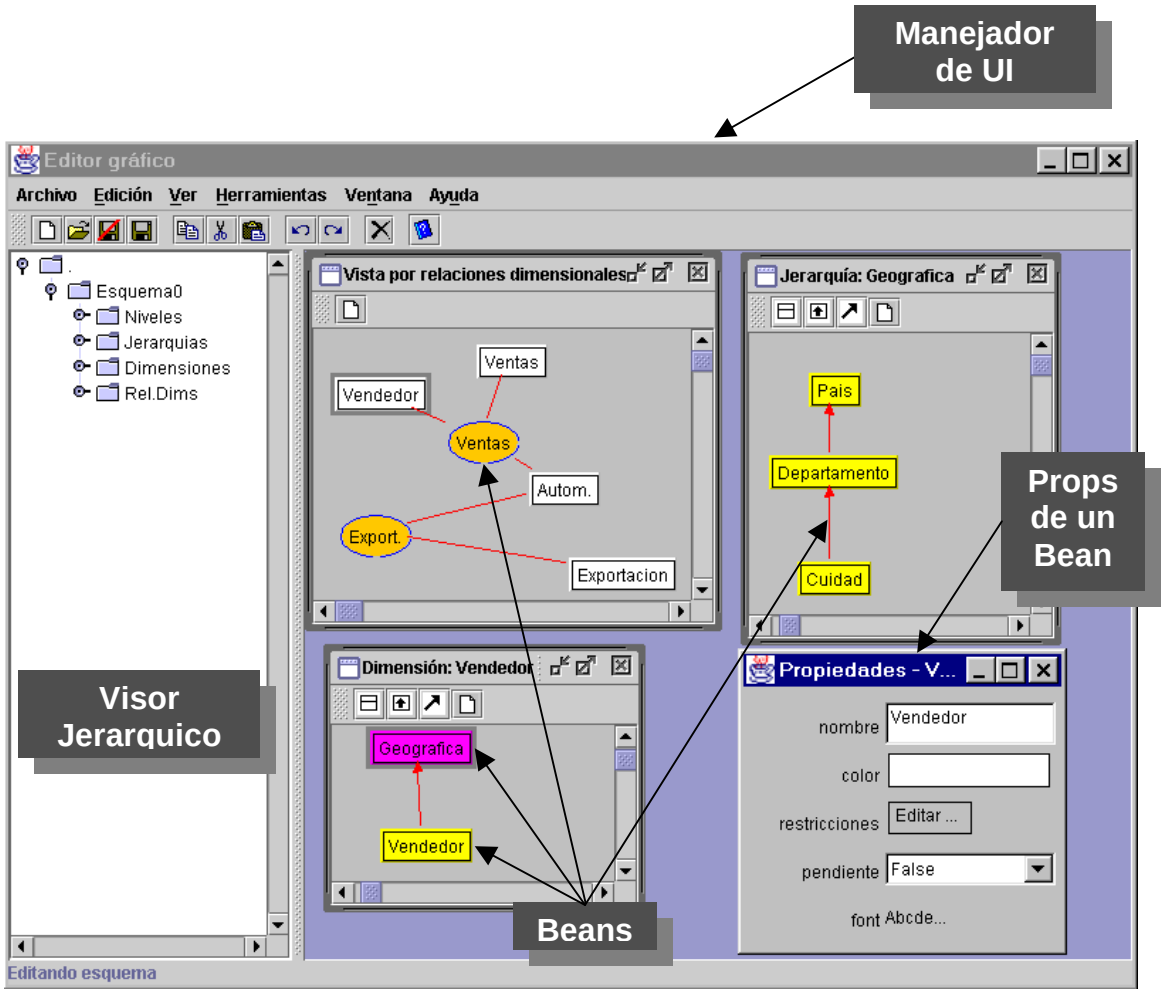

Fig. 42. Ventana del Manejador CMDM.

Actualmente, un nuevo grupo de estudiantes, está completando las funciones del *Manejador de IU*, del *Manejador del Modelo*, y del *Manejador de Soporte,* en el marco de su proyecto de grado. El prototipo del *Manejador del Lenguaje* fue programado en el desarrollo del presente trabajo.

5.3.2. Extensiones a CMDM.

Las *Jerarquías Macro* son estructuras similares a las dimensiones pero que no proveen nueva semántica. Sólo organizan niveles y puede utilizarse en varias dimensiones. La herramienta permite ver la estructura de esta jerarquía o colapsarla en un nivel. En Fig. 42, se puede ver la jerarquía *geográfica* colapsada en la dimensión *Vendedor* y expandida en su propia ventana.

Estructura Pendiente, es un estado que tiene todo objeto del esquema. El mismo está representado por una propiedad booleana en el Bean que representa al objeto. Cualquier objeto nuevo que se agregue, tiene el *estado pendiente* en *true*. Esto indica que aún no está terminado desde el punto de vista del diseño. Una vez que el diseñador considera que el trabajo sobre ese elemento está terminado, debe cambiar el *estado pendiente* a *false*. En base a este estado, el diseñador puede comenzar con una parte del modelo, luego seguir con otra sin haber terminado la primera y luego consultar que elementos del modelo aún tienen el diseño incompleto.

5.4. Comentarios sobre la Implementación.

Entre los puntos más relevantes de la implementación actual, se encuentra la arquitectura del *Manejador CMDM*. El hecho de proveer una interfase general que utiliza JavaBeans, permite que el módulo sea extensible con relativa facilidad.

Entre las extensiones a realizar hay que agregar la edición gráfica de las restricciones standard y el cálculo de los caminos de Drill-Across entre niveles.

Por otro lado, aún no están totalmente resueltos los chequeos que se pueden realizar en el *Manejador del Modelo*. Lo ideal, sería que se pudiera chequear la consistencia del modelo antes de generar código, aunque fuera a costa de limitar un poco las expresiones del lenguaje.

Capítulo 6. Conclusiones y Trabajos Futuros.

En el presente trabajo se definió un modelo de datos conceptual adecuado para construir especificaciones de bases de datos multidimensionales.

El mismo presenta algunas ventajas y desventajas con respecto a los otros modelos analizados en la sección 2.2.

También se propone la construcción de una herramienta CASE basada en el modelo.

En las siguientes secciones se presentan las ventajas y desventajas del modelo comparándolo con los demás modelos y con los requerimientos de la sección 3.1.1.2 y además, se sugieren mejoras para CMDM y la herramienta CASE.

6.1. Propiedades de CMDM.

El modelo definido, presenta buenas características desde el punto de vista de los requerimientos para los modelos conceptuales(3.1.1.2):

- **Independencia de la Implementación.** No hay ningún elemento en el modelo que exprese algún tipo de indicación sobre el lenguaje, la aplicación o el sistema sobre el que se debe implementar la base multidimensional especificada en un esquema CMDM.
- **Abstracción.** Un esquema CMDM representa principalmente elementos estáticos de la realidad, pero con una visión multidimensional. La especificación de los Roll-up, tal vez pueda interpretarse con una visión dinámica, como eventos que se generan al realizar un drill-up en una dimensión y disparan cálculos al ser capturados por un cubo. De esta forma, sólo se representan características estáticas o dinámicas de la realidad, si tener en cuenta aspectos de implementación.
- **Formalidad.** La semántica de las descripciones en Cmdm está formalizada a través de las definiciones de sus estructuras (4.5) y del lenguaje de restricciones (4.4).
- **Constructibilidad.** Se ha hecho un esfuerzo importante en lograr un lenguaje gráfico que permita la especificación de las estructuras y las restricciones. La utilización intensiva de macros facilita tanto la escritura como la legibilidad, y por lo tanto la comprensión, de las especificaciones. Los diagramas son capaces de manejar algunas propiedades en forma relativamente natural, como la dimensionalidad genérica.
- **Facilidad de Análisis.** Las especificaciones no son ambiguas, dada la definición del modelo. Por otro lado, al estar basado en la lógica, el modelo soporta naturalmente una noción de consistencia.
- **Trazabilidad.** Con respecto a este punto, no hay nada específico en el modelo. Sin embargo, en el prototipo existe la noción de “estado pendiente” que apunta a alguna facilidad en este sentido. Por otro lado, es posible extender el prototipo implementando manejo de versiones.

- **Ejecutabilidad.** Se espera que la herramienta CASE sea capaz de generar código para diversas aplicaciones.

Desde el punto de vista Olap, el modelo presenta las tres características mencionadas en la sección 3.1.2:

- **Visión Multidimensional de los Datos.** Esta propiedad está dada por el carácter multidimensional de las estructuras de datos propuestas.
- **Dimensionalidad Genérica.** Para lograr esto, CMDM permite la especificación de cubos en los que no se identifica ninguna medida. Por lo tanto, en esos casos, cualquier dimensión puede cumplir la función de medida.
- **Resumibilidad.** Es posible, especificar las propiedades de resumibilidad de una medida.

Con respecto al proceso de diseño, las relaciones dimensionales proveen un mecanismo que permite la especificación incremental de los cubos necesarios para la descripción de la realidad. En un momento se pueden tomar determinadas restricciones y luego, ir refinándolas por otras más restrictivas. Sin embargo, sería bueno incluir mecanismos de refinamiento dentro del modelo.

Por otro lado, en principio, el modelo no obliga a utilizar ninguna metodología específica. Se puede pensar en construir las especificaciones CMDM, tanto en forma bottom-up como top-down.

6.2. CMDM vs. otros Modelos.

Para comparar CMDM con los otros modelos, en la 0 se presenta una versión completa de la tabla que se presenta en el Anexo I en la Fig. 17:

Modelo	**Tipo de Modelo**	**Indep. del Relacional**	**Def. de Dim. Jerárquicas.**	**Dimen. Genérica**	**Resum.**	**Leng. para rest.**
Li	*Consulta*	*No*	*Relaciones*	*No*	*No*	*No*
Agrawal	*Consulta*	*Si*	*por oper.*	*por oper.*	*No*	*No*
Gyssens	*Consulta*	*No*	*Relaciones*	*por oper.*	*No*	*No expl.*
Cabbibo	*Lógico*	*Si*	*Estructural*	*por oper.*	*No*	*No expl.*
Lehner	*Lógico*	*Si*	*Estructural*	*No*	*Si*	*No*
Sapia	*Concept.*	*Si (Extensión al ER)*	*Estructural*	*No*	*No*	*No expl.*
Golfarelli	*Concept.*	*Si*	*Estructural*	*No*	*Si*	*No expl.*
Franconi	*Concept.*	*Si (Extensión al ER)*	*Estructural (No expl.)*	*No*	*No*	*No expl.*
CMDM	***Concept.***	***Si.***	***Estructural.***	***Si***	***Si (rest.)***	***Si.***

Fig. 43. Una comparación entre modelos multidimensionales.

La primer columna de la tabla identifica al modelo.

La segunda columna clasifica a los modelos en *Orientado a Consulta*, *Lógico* o *Conceptual*.

La tercer columna, indica si el modelo es independiente del Modelo Relacional.

La cuarta columna indica que estrategia se sigue para implementar las jerarquía de las dimensiones.

La quinta columna indica el mecanismo de soporte de la dimensionalidad genérica.

La sexta columna indica si el modelo soporta resumibilidad. En el caso de CMDM se indica que dicho soporte se provee a través de las restricciones de integridad.

La séptima columna indica la existencia o no de un lenguaje de restricciones de integridad. La indicación “No explícitamente” significa que, si bien en los artículos revisados sobre los modelos hay mecanismos que se podrían llegar a adaptar para definir restricciones de integridad, no son manejados en ese sentido.

6.3. Trabajos Futuros.

En esta sección se presentan posibles mejoras y extensiones a realizar tanto en CMDM como en el módulo *Manejador CMDM* de la herramienta CASE.

6.3.1. Mejoras y Extensiones a CMDM.

Algunos de los requerimientos de los modelos conceptuales, se logran mejor que otros. En particular, los requerimientos de independencia de la implementación, abstracción, y formalidad son alcanzados en forma casi optima. Sobre los restantes se pueden hacer las siguiente puntualizaciones:

- **Constructibilidad.** Las especificaciones gráficas debieran servir como medio de comunicación entre las personas involucradas, una vez comprendido el significado de los mismos. Sin embargo, no parece muy realista pensar que los usuarios se adapten a comprender este lenguaje. Debido a esta visión y a pesar del trabajo realizado, el modelo es susceptible de ser mejorado para facilitar la comprensión de los diagramas.
- **Facilidad de Análisis.** A pesar que el modelo soporta en forma natural la noción de consistencia, no está totalmente clara la forma de realizar chequeos efectivos de esta noción. Sin embargo, el hecho de que las bases que se especifican son siempre finitas, abre la posibilidad de que se puedan implementar algunos algoritmos tendientes a facilitar el chequeo de la consistencia. Es posible que para esto se deba restringir el lenguaje de restricciones de alguna forma.

- **Trazabilidad.** Entre las posibles extensiones al modelo con respecto a este punto, posiblemente hubiera que incluir el "estado pendiente" al nivel del modelo y definir un mecanismo que permita llevar adelante en una forma adecuada el control de versiones de los objetos.
- **Ejecutabilidad.** Parece casi imprescindible definir un mecanismos de mapeo sobre otros modelos, en particular sobre el Modelo Relacional.

Con respecto a las propiedades referidas a las aplicaciones Olap, el punto más claro a extender es la **resumibilidad**. Actualmente, el mecanismo actual es directo en el sentido en que se especifican las funciones que se pueden aplicar a un cubo dado. Posiblemente se debiera especificar un mecanismo más cercano a las nociones presentadas en [Len97] y en [Leh98a].

Con respecto al proceso de diseño, sería deseable disponer mecanismos de refinamiento dentro de las estructuras del modelo. Actualmente, el único mecanismo de refinamiento incluido consiste en ir cambiando las restricciones por otras más restrictivas.

6.3.2. Extensiones y Mejoras al Manejador CMDM.

Con respecto al *Manejador CMDM*, existen algunas características que aún no se han implementado y que por lo tanto, deberían integrarse en próximas versiones:

- **Impresión de Diagramas y Generación de Reportes.** En la versión actual no existe ningún mecanismo de impresión del modelo ni tampoco se generan reportes de ningún tipo sobre los componentes. Los dos puntos al respecto a implementarse en la siguiente versión son la impresión de los diagramas y la posibilidad de realizar consultas sobre las propiedades de cada componente para la realización de reportes.
- **Edición Gráfica de Restricciones y Tipos de nivel.** En la versión actual, sólo son strings libres. Debería implementarse al menos, la edición gráfica de restricciones standard.
- **Visualización de los Caminos de Drill-Across por Niveles.** Actualmente, sólo se pueden visualizar por dimensiones. Sin embargo, para lograr esta característica, es necesario basarse en el análisis de las restricciones de integridad para deducir cuáles son los cubos que tienen niveles en común.
- **Implementación de Mecanismos para el Procesamiento del Lenguaje de Restricciones.** En esta versión sólo se analiza el modelo y se crea un repositorio interno, apto para un procesamiento posterior. En la siguiente versión del *Manejador del Lenguaje* se debería incluir operaciones básicas tendientes a facilitar el chequeo de consistencia.
- **Tratamiento de las Jerarquías Macro.** Las jerarquías macro parecen ser una herramienta interesante a la hora de simplificar la presentación de los diagramas. Sin embargo, se deberían permitir otras manipulaciones como la expansión in-situ y la utilización de parámetros.

- **Implementación de Asistentes de Configuración.** La arquitectura propuesta para el *Manejador CMDM*, permite la adaptación del módulo para soportar otros lenguajes gráficos. El módulo *Manejador de IU* es muy parecido a un ambiente de programación basado en JavaBeans. Esto permite que cambiando el resto de los módulos la herramienta se pueda reutilizar en otras aplicaciones. La idea de los Asistentes de Configuración, es que sean programas que generen esquemas de los módulos restantes basándose en algunas información que aporte el usuario.

Bibliografía.

[Agr97] Agrawal, R. Gupta, A. Sarawagi, S.:*"Modelling Multidimensional Databases"*, ICDE, 1997.

[Art99] Artale, A. Franconi, E:*"Introducing Temporal Descriptions Logics."*, ???? DWQ Internet Site, 1999.

[Cab97] Cabibbo,L. Torlone, R.:*"Querying Multidimensional Databases"*, SEDB, 1997.

[Cab98] Cabibbo, L. Torlone, R.:*"A Logical Approach to Multidimensional Databases"*, ", Sixt Int. Conference on Extending Database Technology (EDBT '98), Springer-Verlag, 1998.

[Cab99] Cabibbo, L. Torlone, R.:*"Data Independence in OLAP Systems"*, Dipartimento de Informatica e Automazione, Università di Roma Tre, 1999.

[Cal98] Calvanese, D. Lenzerini, M. Nardi, D.:*"Description Logics for Conceptual Data Modeling."*, Logics for Database and Information Systems., J. Chomicki and G. Saake eds., 1998.

[Car97] Carpani, F. Goyoaga, J. Fernandez, L.:*"Modelos Multidimensionales"*, IV Jornadas de Informática e Investigación Operativa. Instituto de Computación de la Facultad de Ingeniería. Monteviedo, Uruguay., 1997.

[Cod93] Codd, E.F. Codd, S.B. Salley, C.T.:*"Providing OLAP to user-analysts. An IT mandate."*, Technical Report., E.F. Codd and Associates., 1993.

[Fon00] Fontan, Mariana. Picerno, Alejandro.:*"Un CASE para Olap. "*, Informe de Proyecto de Grado., Abr. 1900.

[Fra99] Franconi, E. Sattler, U.*"A Data Warehouse Conceptual Data Model for Multidimensional Aggregation: A Preliminary Report."*, Technical Report., DWQ, 1999.

[Fra99a] Franconi, E. Sattler, U.:*"A Data Warehouse Conceptual Data Model for Multidimensional Aggregation."*, Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99), S. Gatziu, M.Jeusfeld, M.Staudt, Y. Vassiliou, Eds., Jun. 1999.

[Gol98] Golfarelli, M. Rizzi, S.:*"A Methodological Framework for Data Warehouse Design"*, First International Workshop on Data Warehousing and OLAP (DOLAP), ACM, Nov. 1998.

[Gol98a] Golfarelli, M. Maio, D. Rizzi, S.:*"Conceptual Design of Data Warehouses from E/R Schemes."*, International Conference on Systems Science, Hawaii, IEEE, Ene. 1998.

[Gol99] Golfarelli, M. Rizzi, S.:*"Designing the Data Warehouse: Key Steps and Crucial Issues.",* Journal of Computer Science and Information Management. Vol. 2. N. 3, Maximilian Press Publisher, 1999.

[Gou97] Goubault-Larreacq, Jean. Mackie, Ian.:*"Proof Theory and Automated Deduction.",* Book from Applied Logic Series., Kluwer Academic Publishers, 1997.

[Gru99] Grumback, S. Rafanelli, M. Tininini, L.:*"Querying Agregate Data",* Proceedings Conference on Principles of Database Systems (PODS '99), Philadelphia, ACM, 1999.

[Gys97] Gyssens, M. Lakshmanan, L.:*"A Foundation for Multi-Dimensional Databases",* VLDB, Athens, Greece, 1997.

[Hor99] Horrocks, I. Sattler, U. Tobies, S.:*"Practical Reasoning for Expressive Description Logics.",* DWQ Internet Site., 1999.

[Hor99a] Horrocks, I. Sattler, U. Tobies, S.:*"Practical Reasoning for Description Logics with Functional Restrictions, Inverse Roles and Transitive Roles, and Role Hierarchies.",* DWQ Internet Site, 1999.

[Jar99] Jarke, M. Lenzerini, M. Vassiliou, Y. Vassiliadis, P.:*"Fundamentals of Data Warehouses",* , Springer-Verlag, 1999.

[Ken96a] Kenan Technologies:*"An Introduction to Multidimensional Databases.",* White Paper, Kenan Technologies, 1996.

[Kim96] Kimball, Raplh.:*"The Datawarehouse Toolkit.",* Book., John Wiley & Son, Inc., 1996.

[Leh98] Lehner, W. Albrecht, J. Wedekind, H.:*"Normal Forms for Multidimensional Databases.",* , 1998.

[Leh98a] Lehner, W.:*"Modeling Large Scale Olap Scenarios.",* Proc. EDBT '98. Valencia. España., http://www6.informatik.uni-erlangen.de/dept/staff/lehner-publications.html, 1998.

[Len97] Lenz, H. Shoshani, A.:*"Summarizability in OLAP and Statistical Databases.",* 9th Int. Conf. on Statistical and Scientific Databases , 1997.

[Li96] Li, C. Wang, X.S.:*"A Data Model for Supporting On-Line Analytical Processing",* Conf. on Information and Knowledg Management, Nov. 1996.

[Lou92] Loucopoulus, P. Zicari, R.:*"Conceptual Modeling Databases and CASE",* , John Wiley & Sons, Inc., 1992.

[Man99] Mangisengi, O. Tjoa, A M.:*"A Multidimensional Modeling Approach for OLAP within the Framework of the Relational Model based on Quotient Relations",* Proceedings Conference on Principles of Database Systems. (PODS ´99) Philadelphia, ACM, 1999.

[Sap99] Sapia, C. Blaschka, M. Höfling, G.:*"An Overview of Multidimensinal Data Models for OLAP",* Technical Report., http://www.forwiss.de/~system42/publications, Feb. 1999.

[Sap99a] Sapia, C. Blaschka, M. Höfling, G. Dinter, B.:*"Extending the E/R Model for the Multidimensional Paradigm.",* Advances in Database Technologies. LNCS Vol 1552, Springer-Verlag. 1999.

[Sap99b] Sapia, C. Blaschka, M. Höfling, G.:*"GraMMi: Using a Standard Repository Management System to Build a Generic Graphical Modeling Tool",* ???, ???, +Ju. 1999.

[Tho97] Thomsen, E.:*"OLAP Solutions. Building Multidimensional Information.",* Book, John Wiley & Sons, Inc. , 1997.

ANEXO I.

Multidimensional Models: A State of Art.

Fernando Carpani

1. Introduction.

In the last four years, some multidimensional models were defined. Some of these models are focused on query specifications, but only a little has really a conceptual approach.

The next section is an attempt to present some fundamentals over some of "query models". These models are summarized in [Car97] and [Sap99]. For such models, only some comparisons are made with a special emphasis on data structures.

The following sections are dedicated to some relevant works on Multidimensional Conceptual and Logical Modeling ([Cab97], [Gol98] , [Fra99a] , [Sap99a]).

Later, some conclusions about the works are presented.

To read the next sections, some considerations must be assumed:

- The basic structure on a multidimensional model is the hypercube or cube, a multidimensional array with discrete valued dimensions.
- These dimensions are hierarchically structured and some basic operations allow navigate across them.
- The navigation over dimensions can make a different view, or a transformation of the original cube. Typically, these transformations have an aggregation operation related to.

2. The "Query" Models.

These models are focused on describe data manipulation operations and not on specify a data structure. However, they must be based on some data structure to express these manipulations.

The data structure is, basically, the same:

> *A Multidimensional Database is a set of Cubes built on some hierarchically structured dimensions with a set of discrete values on each level of each dimension*. *A Cube is a partial function from the Cartesian Product of levels from different dimensions in other set of values called Measures*.

The major difference between these models is the way this structure is represented.

2.1. Li and Wang.

For Li and Wang ([Li96]), a cube schema is a set of pairs (D_n,R_n) where D_n is a dimension name and R_n is a set of attribute names (relation schema, similar to Relational Model).

In this model, a cube instance is a pair (F, μ) where F is a set of pairs (D_n,r_n) and μ is a function. The pairs in F are composed by a dimension name that is the first element of some pair in the cube schema, and a relation for the attribute set associated to the dimension name in the cube schema. The second element of the pair μ is a function which associate a scalar value in the set ν to each value in the set $\{\{(D_1,t_1,),...,(D_n,t_n)\} \mid \forall(i\in[1,n]).(ti \in ri)\}$. This set can be seen as the set of all n-tuples where each tuple is composed by each $t_i \in r_i$ in F.

An advantage of this point of view is that a cube is a function from dimensions into some values.

The queries in this model can be made with a language named *grouping algebra*. This language is a Relational Algebra extension with notions similar to *order by* and *group by* in SQL. This makes the model Relational dependent, i.e. it is too oriented to Relational Data warehouses.

The formalizations appear as too intricate to specify an SQL-like language. This model can be seen as a specification of *Star Schema [Kim96]* .

2.2. Gyssens & Lakshmanan.

The model presented in [Gys97] is based in one basic structure that is a representation of a cube: the table.

Like the previous one, this model is highly based in the Relational Model but the structure is softly different.

A *table schema* is a triplet *<D,R,Pair>* where:

- $D = \{d_1,\ldots,d_n\}$ is a set of dimension names.
- $R = \{A_1,\ldots,A_n\}$ is a set of attributes.
- Pair: $D \rightarrow 2^{\{A_1,\ldots,A_n\}}$ such
  1) $\forall(<i,j>\in[1,n]X[1,n])(i\neq j \rightarrow (Pair(d_i) \cap Pair(d_j) = \varnothing))$
  2) $\cup_{d\in D}Pair(d) \subseteq R$

This structure is similar to the one used by the previous proposal, but has a different way to associate a dimension name with a set of attributes. While Li and Wang use a list of pairs, this proposal uses a function (*Pair*) from dimension names into sets of attributes names with some constraints over the dimensions: the dimensions must be disjoint.

In this proposal, a cube can have a set of measures. The measures are attributes from $R - \cup_{d\in D}Pair(d)$.

An *instance* for a table is defined as n+1 finite relations where the first n relations are over the attributes from each dimension with an extra attribute for tuple identification (T_{id}). The last relation is over the attributes from M and has each one (T_{id}) from the first n relations. With a simple constraint, this can be seen as a specification for *Star Schema* too. This constraint is:

$$\pi_{Tid}(r_{d1}) \otimes \ldots \otimes \pi_{Tid}(r_{dn}) = \pi_{rd1.Tid,\ldots,rdn.Tid}(r_m) \ (\otimes \text{ natural join})$$

In this model, the operations are presented as a relational algebra extension and have some operations oriented to representation. A table has two representations: One tabular and one relational (see Fig. 1 and Fig. 2). This can be considered as an advantage because simplify the algebra definition. When an operation is inherited from Relational Algebra, is defined as the same operation in the relational representation. This example was taken from [Gys97] .

PRODUCTION			Time						
			Year	1996		...	1997		
			Month	Jan.	Feb.	...	Jan.	Feb.	
	Product	Employee	(Prod. / hour, Garbage)						
CAT.	Pistons	John Smith		(5,6)	(5,7)	...	(4,6)	(4,8)	...
		Joe Jones		(5,7)	(5,8)	...	(4,8)	(4,9)	
		...		...	...		...	...	
	Valves	John Smith		(7,8)	(7,9)	...	(6,9)	(6,8)	
		Joe Jones		(6,9)	(6,9)	...	(5,8)	(5,9)	
		...		...	...	...	...	...	
	...	...		...	...	...	...	...	

Fig. 1. A multidimensional representation of a table.

Category

Tid	***Part***	***City***
c1	Pistons	J. Smith
c1	Pistons	J. Jones
...	...	...
c6	Valves	J. Smith
c7	Valves	J. Jones

Time

Tid	***Year***	***Month***
t1	1996	Jan.
t2	1996	Feb.
...	...	...
t13	1997	Jan.
t14	1997	Feb.

rm

C.tid	***T.tid***	***Prod/hour***	***Garb.***
c1	t1	5	6
c1	t2	5	7
...	...	...	...
c6	t13	6	9
c6	t14	6	8

Fig. 2. A relation representation of a table.

The relational algebra is extended with aggregate functions and “group by” mechanisms.

There are two functions for restructuring tables: Fold to transform a dimension attributes into measures, and Unfold to transform measures into dimension attributes.

These operations are oriented to implement the symmetry between dimensions and measures. ([Cod93])

In summary, this model can be seen as a more simple specification for Star Schema than the previous model. Also has some advantage as the symmetry between dimensions and measure. However, still is Relational Dependent.

2.3. Agrawal, Gupta and Sarawagi.

The model is presented by the authors as a logic model, as an attempt to uniform and extend the multidimensional database functionalities. The model yields on some geometric notions.

The basic structure is the *Hipercube*. A hipercube has two elements:

- Dimensions. A dimension is a name d_i and an associate domain dom_i.
- Elements (or measures). This is defined as a map

$$E(C): dom_1 X \ldots X dom_n \rightarrow \text{n-tuples} \cup \{0,1\}$$

The set n-tuples are tuples of measures.

In this model, a coordinate can be mapped to a Boolean or to a tuple. If the result of such application is a tuple, then must be a tuple with the same structure for each other coordinate. If the result is a Boolean then must be a Boolean to each other coordinate. In this way, a hypercube can be seen as a Boolean function and with a process similar to functional curryfication dimensions can be transformed in measures.

An algebra is defined over the model. This algebra is not directly relational algebra. It has different operations.

Some operators are:

- *Push*. Transform a dimension in a measure.
- *Pull*. Transform a measure in a dimension.
- *Destroy Dimension*. Delete a hypercube dimension. The dimension to delete only can have one value.
- *Restriction*. Delete each value in the hypercube whose respective values in a specific dimension does not verify certain condition.
- *Join*. With two hipercubes, construct a new hypercube. Each dimension in the first hypercube is combined with only one dimension of the second hypercube. The result is a dimension with the union of the original dimensions. The measures from each cube for each coordinate are combined in a new measure, as result of an operation f_{elem} .

In summary, this model has a great generality degree, based in a functional view of multidimensional structures and operations. The major features of the model are summarized in the following items:

- There is not an explicit difference between schema and instance of a structure.

- This model has support for symmetry between measures and dimensions. To this introduces a "Boolean cube" which is used by other models
- The model can be mapped to SQL in an easily way. Therefore, this model is valid in a ROLAP environment, but it is not exclusive. The model can be used on MOLAP or HOLAP environments to.
- There is an explicit operation to relate two hipercubes.
- There is not an structural way to represent hierarchies. These are managed with auxiliary functions like f_{merge} or f_{elem}.
- This model is presented by the authors as a logical model so, it is oriented to express manipulations.

3. Logical and Conceptual Models.

In this section, the main models with conceptual features are sketched. These models will be presented deeper than the previous ones in order to expose in the future the differences with CMDM.

The assumptions presented in the introduction are also valid with these models and the basic multidimensional structure underlying are the same as the previous.

3.1. Cabibbo and Torlone.

Their proposal is based on a logical model named *MD*. This work can be read in three papers: Querying Multidimensional Databases ([Cab97]) , A Logical Approach to Multidimensional Databases ([Cab98]) and Data Independence in Olap Systems ([Cab99]).

The first work, presents a logical multidimensional model with a clear multidimensional structure, a calculus as query language, and some results on query languages expressive power.

The second work is oriented to logical design on OLAP and is based on the model presented in the first work. Here, some design methodology is presented. This methodology is oriented to transform an ER schema of operative data into a MD schema.

In the third work, an architecture for Olap Systems is presented introducing the Data Independence notion on these systems.

In this section, the first version of MD model is presented. This is in the assumption that this version of MD is the major advantage of their proposal.

This work can be considered as a "Query Model", but it's recognized by its authors as a logical model and, in [Cab98] its conceptual features are exposed.

3.1.1. Data Structures.

There are two fundamental structures in the model: the *dimension* and the *f-table*. The last one is used to represent the cube notion.

A multidimensional database is a pair with a set of dimensions and a set of f-tables constructed over those dimensions.

A set of names £ is assumed. Each name in this set is called *level*. This set has two conditions:

- For each $l \in$ £, exists a set of values DOM(l). These sets are called *level domains*.
- For each $l \in$ £ and for each $l' \in$ £, $DOM(l) \cap DOM(l') \equiv \varnothing$.

The last condition express that every pair of level domains are disjoint.

3.1.1.1. Schemas.

Formally, a *dimension schema* is a triplet **<L,≤,R-UP>** where:

- L is a set such: $L \subseteq$ £ and L is finite. This is a finite set of level names.
- ≤ is a partial order over L. When $l_1 \leq l_2$ you can say that l_1 *rolls-up to* l_2 or that l_2 drills-down to l_1. This is a partial order over the levels and represents the dimension hierarchy.
- R-UP is a family of functions: the roll-up functions. These functions describe the way the navigation over the dimension hierarchy from an element level to another one takes place.

An f-table schema has the following form:

$$\mathbf{f\ [\ A_1:\ l_1,....,A_n : l_n\ ] : l_0}$$

Each l_i is a level from any dimension. The symbol *f* is the schema name and each A_i is an attribute of *f*. The level $\mathbf{l_0}$ is called a *measure*.

As were presented above, a multidimensional schema is a pair **<D,F>** where:

- D is a set of dimensions.
- F is a set of f-tables such each l_i in each f-table in this set, is a level from some dimension of D.

3.1.1.2. Instances.

A *symbolic coordinate* over a f-table $f[\ A_1:l_1,....,A_n:l_n\ ]:l_0$, is a function which maps each attribute name A_i in an element of $DOM(l_i)$ where $1 \leq i \leq n$. This definition is congruent with the classical notion of *tuple*.

An instance of an f-table $f[\ A_1:l_1,....,A_n:l_n\ ]:l_0$ is a function coordinates over f in $DOM(l_0)$. This function is defined over a finite set of coordinates over f.

An instance of a multidimensional schema S=<D,F> is a function which maps each f-table schema in F to an f-table instance.

3.1.1.3. Other Features.

A family of query languages is defined over these structures under the form of a parametric multidimensional calculus. The generic form of a query is the following:

$$\{\ x_1,...,x_n{:}x \mid \psi(x,x_1,...,x_n)\}$$

In this expression, x_i are variables and x is a distinguished variable called *result variable*. The expression denoted with ψ is a first order formula involving roll-up expressions, scalar functions and aggregate functions. The scalar functions can be user or system provided. Based on this feature, the language can be oriented to different applications.

With two query examples, the application of roll-up and f-table instances can be saw.

The f-table *production* represents the diary units produced, classified by product and employee.

Production(*day:*day, *item:*product_id, *name:*employee_name): numeric

1. To represent the diary units from each product by the employee with name Smith, the following query can be used to build a new f-table:

$$SMITH\_UNITS = \{ x_1, x_2, x_3 : x \mid x = Production[day: x_1, item:x_2, name:x_3] \wedge \text{“SMITH”} = R\text{-}UP_{employee\_id}^{name}(x_3) \}$$

2. To define a new f-table to represent the summary of production in a week of each item and in each factory area, the following query can be used:

$$PRODUCTION\_SUMMARY = \{ x_1, x_2, x_3 : x \mid x = sum (y_1, y_2 : y \mid y=Production[day: y_1, item:x_2, name:y_2] \wedge x_1= R\text{-}UP_{day}^{week}(y_1) \wedge x_3= R\text{-}UP_{employee\_id}^{area}(y_2)) \}$$

This query language allows defining a new Boolean f-table from any previous f-table in a practical way. This feature can be used to express the symmetry between dimensions and measures. These Boolean f-tables are called by the authors, *abstractions*.

3.1.1.4. Conclusions.

The model represents the basic data structures in a direct way. This is an advantage to build a conceptual model over it.

The roll-up paths are represented in a structural way. Because of this fact, an increment of precision in the analyst descriptions may be raised.

In [Cab98], the version of the model includes a specification of *descriptions* that are descriptive attributes.

The feeling of these papers is that the authors think that a level must be scalar or string. This constraint is not explained in nowhere.

CMDM is based on this model, proposing some changes and extensions.

3.2. Lehner.

In [Leh98a] presents the “Nested Multidimensional Model”.

This model is oriented to operations, but implements multidimensional structures in a direct way. It can be considered as a logical model.

3.2.1. Structures.

The dimensional structures are based on the fact that the attributes are classified in the following three categories:

- **Primary attributes (PA)**. It identifies the *dimensional elements*, i.e. the elements of a dimension.
- **Classification attributes (CA)**. It allows structuring the elements of the dimension in levels. These attributes describe a balanced tree called *classification hierarchy* where each node is an instance of the classification attribute for that level.
- **Dimensional attributes (DA)**. It describes features of the dimensional elements.

The classification hierarchy has a value for each classification attribute in its internal nodes and a value for the primary attribute for the dimension in its leaves. The root always is the level *top* with the node *all* that must be considered as a classification attribute. In Fig. 3 can be seeing the different attributes for a dimension.

For de root and for each internal node, are defined two domain types:

- **Classification Oriented Domain (DOM($C_{|CA}$))** contains the values of each classification or basic object from the level CA that belongs to the sub-tree rooted on the node C.
- **Feature (or Description) Oriented Domain (DOM($C_{|DA}$))** contains the values of the attribute DA for each node that belongs to the sub-tree rooted on the node C.

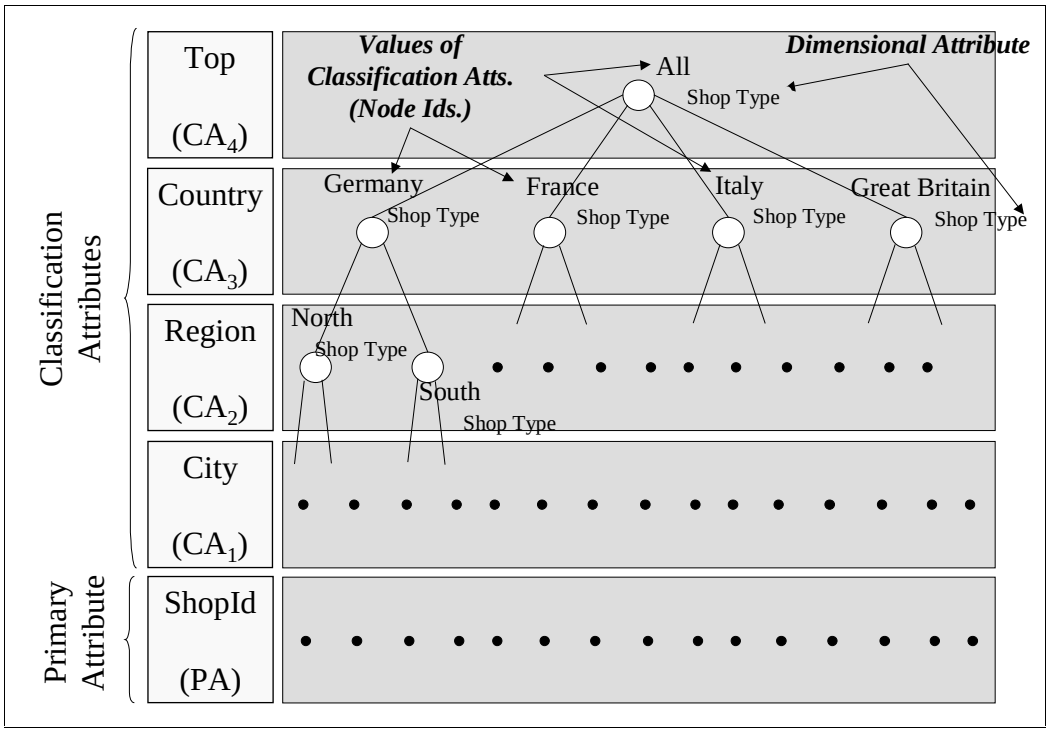

Fig. 3. Node Domain an attribute Identification for a Shop dimension.

The expression *DOM (Germany|$_{Region}$)* denotes the classification domain for the node *Germany* with respect to the level (or classification attribute) *Region*. This domain contains only the values *North* and *South*.

The expression *DOM(Germany|$_{ShopType}$)* denotes the description domain for the node *Germany* with respect to the *Shop Type* description attribute. The values contained in this domain are the values for the attribute *Shop Type* for the sub-tree rooted on this domain, so, must contain the values of *DOM(North$_{|ShopType}$)* and *DOM(South$_{|ShopType}$)*.

The attribute *Shop Type* is on all nodes but is not a rule. A dimensional attribute must be in a context (sub-tree) depending on the value of the node. In [Leh98a] there is an example where exists a Product dimension. Some products are Video Recorders, Washers and Dryers. The Video Recorders can have an attribute Video System (pal, ntsc, etc.) which no sense for Washers and Dryers. This dimensional attribute only appears in the sub-tree for Video Recorders.

Over this notions, the author defines three types of multidimensional objects:

- **Primary Multidimensional Objects (PMO)** are tuples with five items: a selection item (context descriptor), an aggregation item (context descriptor schema), an aggregation operation type, a data type and a cell identifier. The selection item is a tuple of node identifiers, i.e. a tuple of classification attributes values. The aggregation item is a tuple of classification attributes, which match with the values in the selection. The aggregation operation type can be summarization, average or a constant. The data type can be natural, integer or real. The cell identifier is only a name for the structure. Note that in any of the previous items are dimensional attributes.
- **Secondary Multidimensional Objects (SMO)** are pairs with a context descriptor and a set of dimensional attributes. This set of dimensional attributes must be a subset of the union of the dimensional attributes of each node included in the context descriptor.
- **Multidimensional Objects (MO)** are pairs with a PMO and set of dimensional attributes. This dimensional attributes are used for define a set of SMOs associated with the PMO.

The PMOs defines the dimensional structure of a cube. The SMOs adds the measures using the contextual information of the PMO. The MOs are really cubes with a SMO for each element in the PMO.

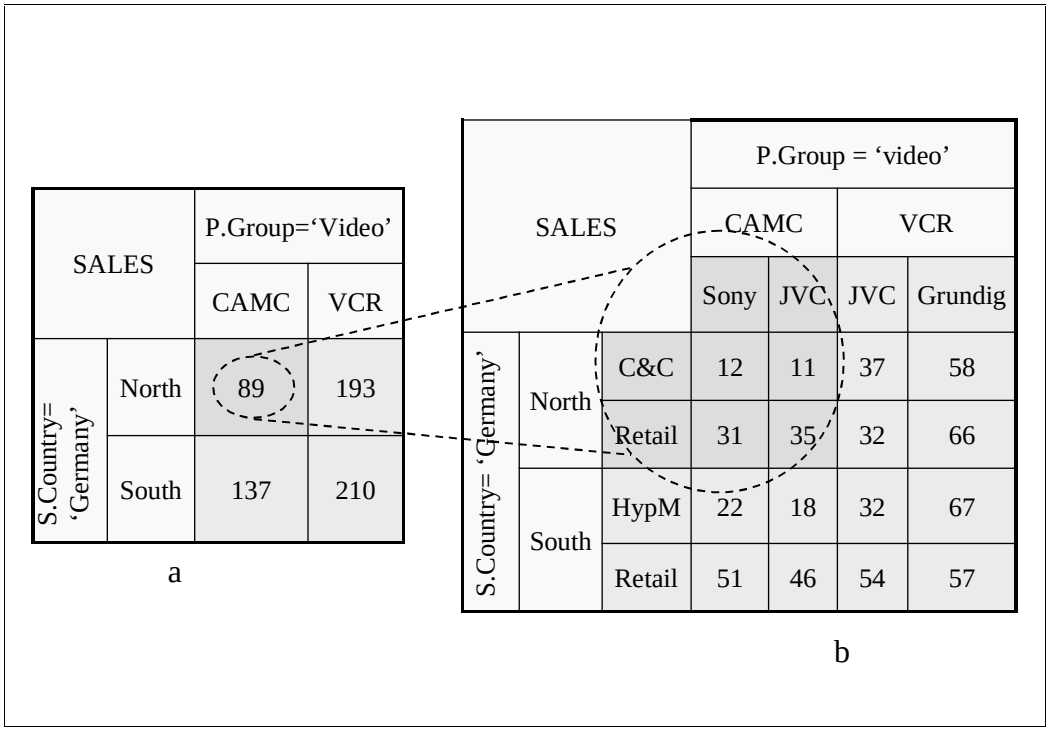

Fig. 4. Sample MOs in a tabular representation showing context-sensitive nesting.

The primary and secondary multidimensional objects have domains formally defined. The domain of a PMO is the Cartesian product of the classification-oriented domain for each element in the context descriptor.

In the table **a** in Fig. 4, the clearest cells are the PMO. The first cell is the *cell identifier*, the first column and line, which have conditions, are the context descriptor and the second column and line have the elements of the classification-oriented domain for each element from the context descriptor. The rest of the table is a representation of the cartesian product of the domains of the context descriptor. Note that the lines rises of a Shop dimension and the columns rises of a Product dimension.

The figure represents a MO so; in each element of this Cartesian product, there is a SMO. The domain of a SMO is the cartesian product of the description-oriented domain of the elements in the context descriptor with respect to the dimensional attributes in the SMO. In the table **b**, each SMO is presented as a inner table (darker shaded) for each element of the PMO.

The darkest zone in the tables shows a specific SMO. In the **a** table, this SMO is 0-dimensional, but in the **b** table is 2-dimensional.

Note that in the PMO there is an aggregation operation type not an aggregation function. If the aggregation operation type is summarization, the function can be SUM, AVG, MIN, MAX or COUNT. If the aggregation operation type is average, the function can be AVG, MAX or MIN. If the aggregation type is constant, there is no function to apply.

In the example can be assumed that the operation is count.

In the rest of the paper, an algebra is defined, with the natural multidimensional operation included (roll-up, slice, etc.)

3.2.2. Conclusions.

The model must be considered as a logical model in spite of being query oriented.

In spite of the model is query oriented, must be considered as a logical model because expose clearly the multidimensional structures and operations.

In the analyzed models, only this one has some notions of summarizability.

The model seems to be adequate to express dynamic cubes. This feature rises from the context definitions.

3.3. Golfarelli, Rizzi et. al.

In [Gol98a] the authors present a multidimensional data model called Dimensional Fact Model (DFM.). A method to construct a multidimensional schema in this model from an E/R schema is also presented. This model is presented as graphical notation without any formalization.

In [Gol98] and in [Gol99] there is a reformulation of the model focused on a design methodology and it formalization is presented.

Since this model is focused on conceptual design, the emphasis is on the structures and not on operations.

3.3.1. Structures.

3.3.1.1. Graphical Model.

The basic structure of the model is the *fact scheme*. A graphical representation can be saw Fig. 5. This example was taken from [Gol98a].

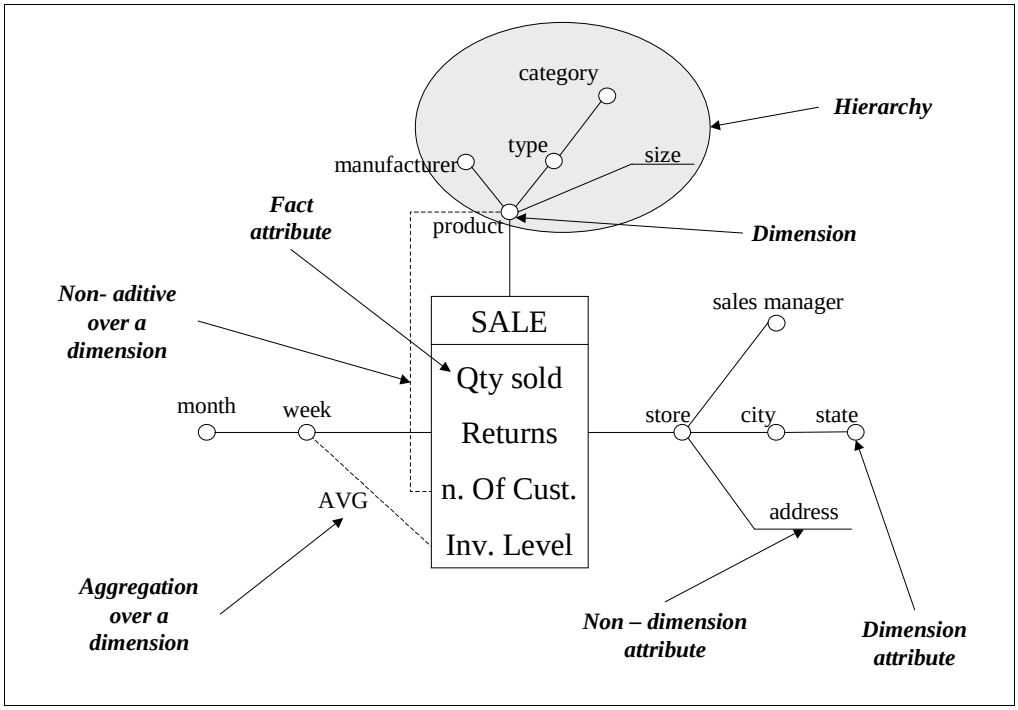

Fig. 5. A Fact Scheme.

3.3.1.2. Formal Model.

This structure is formalized using a kind of Directed Acyclic Graph (DAG) called *quasi-tree*. If g=(V,E) is a directed, acyclic and weakly connected graph, then g is a *quasi-tree* with root v_0, if all vertexes $v_j \in V$ must be reached from v_0 through at least one directed path.

A fact scheme is a tuple

f=(M,A,N,R,O,S)

where:

- M is a set of numerical or Boolean expressions involving values from the operational systems. Each $m_i \in M$ is called a *measure*.
- A is a set of identifiers called *dimension attributes*. Each $a_i \in A$ has a discrete domain *Dom(a_i)* associated with it.
- N is a set of identifiers called *non-dimension attributes*.
- R is a set of ordered pairs. Each pair (a_i, a_j) verify that $a_i \in A \cup \{a_0\}$ and $a_j \in A \cup N$ and $a_i \neq a_j$ and *qt(f)* is a quasi-tree with root a_0 where qt(f) is the following graph:

  qt(f)=($A \cup N \cup \{a_0\}$,R)

  The dummy vertex a_0 represent the facts.

- $O \subset R$ is a set of optional relationships.
- S is a set of *aggregation statements*. Each element of this set is a triplet (m_j,d_i,Ω) where m_j is a measure, $d_i \in Dim(f)$ and Ω is an aggregation operator. If $(m_j,d_i,\Omega) \in S$ then the measure m_j can be aggregated over the dimension d_i using the operator Ω. The expression Dim(f) can be defined as:

$$Dim(f)=\{a_i \in A \mid \exists(a_0,a_i) \in R \}$$

If $d \in Dim(f)$ then d is a *Dimension*. Therefore, a dimension is an attribute directly connected to the facts.

A *hierarchy* over the dimension d_i is the quasi-tree $sub(d_i)$. In this way, the hierarchy over a dimension is the graph of its attributes.

3.3.2. Instances

The instances are defined in two steps.

- In a first step, the *Primary Fact Instances* are defined. This is no other thing that the coordinates: a tuple with one value from each dimension domain. Each primary fact instances describe one value for each fact attribute (measures).
- In the second step, the *Secondary Fact Instances* are defined. These instances are aggregations of primary fact instances. These definitions are based on legal *v-dimensional aggregation patterns* which are sets of dimension attributes such there is no path between two of them.

The formalization of these concepts is hard to be present here. It can be found in [Gol98] and [Gol99].

3.3.3. More over the Model.

3.3.3.1. Drill-Across Paths.

The model is oriented to represent independent fact schemes. For that reason, there is no a clear way to represent the drill-across paths.

To solve this drawback, a mechanism for the fact schemes integration is proposed. This mechanism can be sketch out as the union of the measures and the intersection of the dimensions. In this way, if a designer needs to specify drill-across paths, then he/she must integrate the two fact schemes involved in the drill-across.

3.3.3.2. Query and Workload.

The methodology presented in [Gol98] take in account the workload in the data warehouse. So:

- A simple language to express queries is defined.
- The Workload on a dimensional scheme is defined as a set of pairs $<q_i,v_i>$ where q_i is a query and v_i is it expected frequency.

3.3.4. Conclusions.

The formal model is a formalization of the graphical language taking in account some additional considerations. However, it's not a direct formal representation of the multidimensional concepts.

The model doesn't support generic dimensionality and the way to represent the drill-across paths is tricky. However, this method can be applied in a more cases than an explicit drill-across because it can be used for schema integration.

In the notion of *legal aggregation pattern*, there is another underlying notion: Independence between dimensions or at least, levels. This notion is present in the meaning of the term "multidimensional", but only a few times is explicit like here.

Some lines over workload and data volume treatment are presented. Also is presented a translation method to star schema. However, there's nothing about multidimensional DBMS.

This work presents a complete design methodology for DW, which goes from a conceptual to a physical (but relational only) model.

3.4. Sapia, Blaschska et al. (System 42).

In [Sap99a] a multidimensional extension to E/R model called ME/R (Multidimensional Entity Relationship) is presented. In the paper, there is a lot of "common sense" in the reasons for extend E/R model.

The extension is a specialization of E/R model, based on the ISO/IRDS standard metadata. So, only the structures are described. The semantic is similar to E/R.

3.4.1. Structures.

In ISO/IRDS, a meta-model is used to describe a model. This meta-model is similar to E/R and the authors use an extended version with a generalization concept. This model is different from that whose extended version will be the ME/R. This last model is our every day E/R.

In summary, three models are in this discussion and must not be confused:

- The meta-model, which is an E/R model with generalization.
- The E/R model, where the modeler decides which constructions are allowed.
- The Multidimensional E/R, which rises from the previous E/R as a specialization.

The ME/R model has three specializations respect to E/R:

A special entity set: dimension level.

A special n-ary relationship set: the "*fact*" relationship set.

A special binary relationship set: the "*rolls-up to*" relationship set.

The meta-model of ME/R can be saw in the Fig. 6. The gray zone is the extensions added to E/R. The rest is the meta-model of the basic E/R.

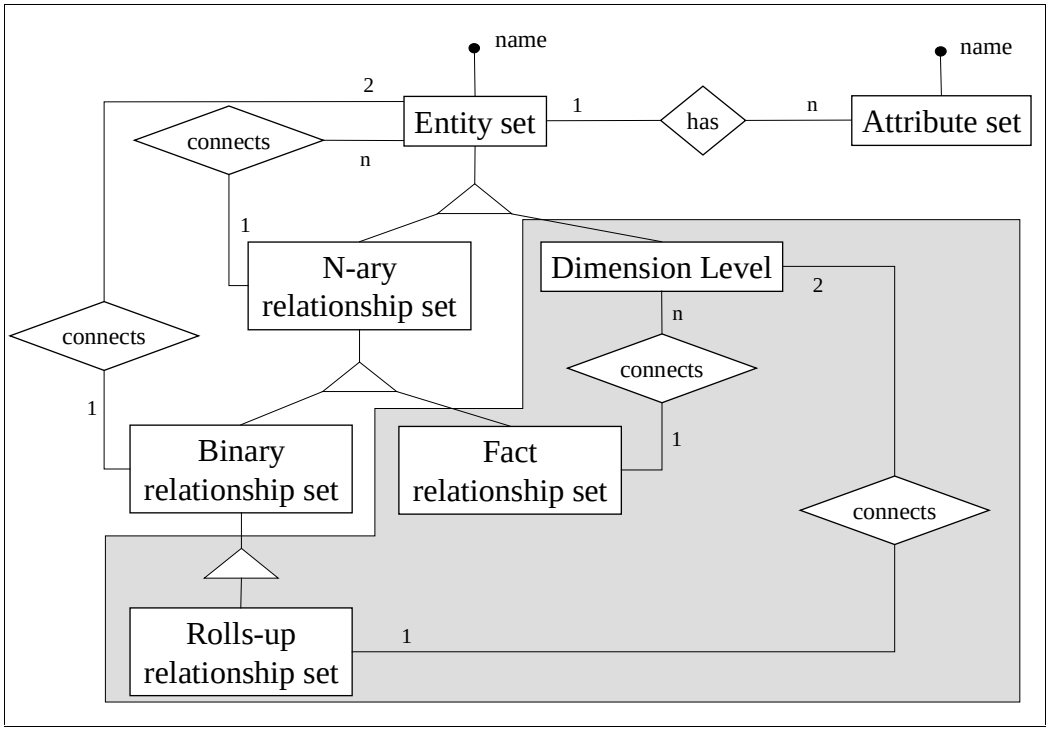

Fig. 6. The meta-model of the ME/R.

A *Dimension level set* is an entity set. A *Fact Relationship Set* is a n-ary relationship set that connects n dimension levels. Dimension attributes are modeled as elements of dimension level attribute set. Measures are modeled as elements of fact relationship attributes set. A *Rolls-up Relationship Set* is a relation between dimension levels. The relationship *connects* between a Rolls-up relationship set and a dimension level has a constraint: the graph constructed with the pairs of dimension levels connected by a Rolls-up relationship Set must be a DAG (Directed Acyclic Graph).

3.4.2. Graphic Notation and Example.

Each extension has a graphic notation related to. These notations can be seen in the Fig. 7.

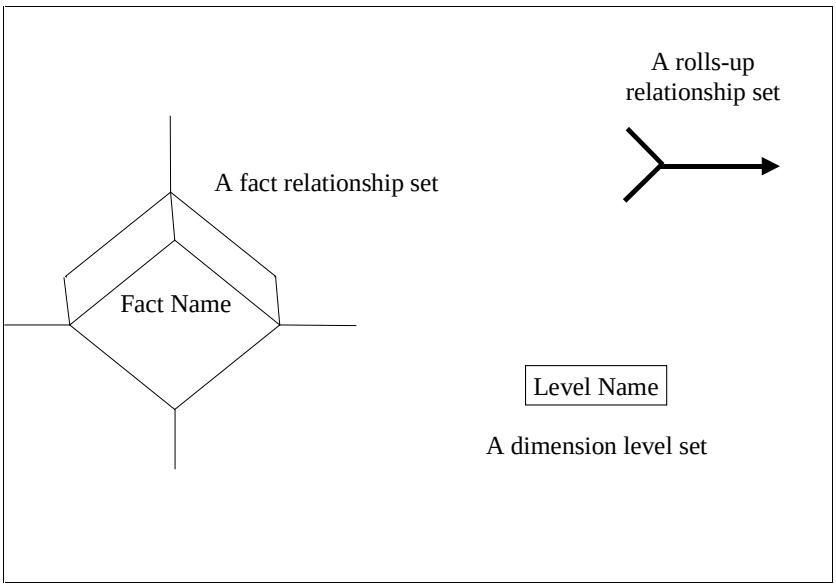

Fig. 7. ME/R graphic extensión to E/R.

These components can be combined freely as all constructions in E/R. There is only a constraint presented above: the graph of dimension levels cannot be cyclic.

The next figure presents the same example as in the previous model.

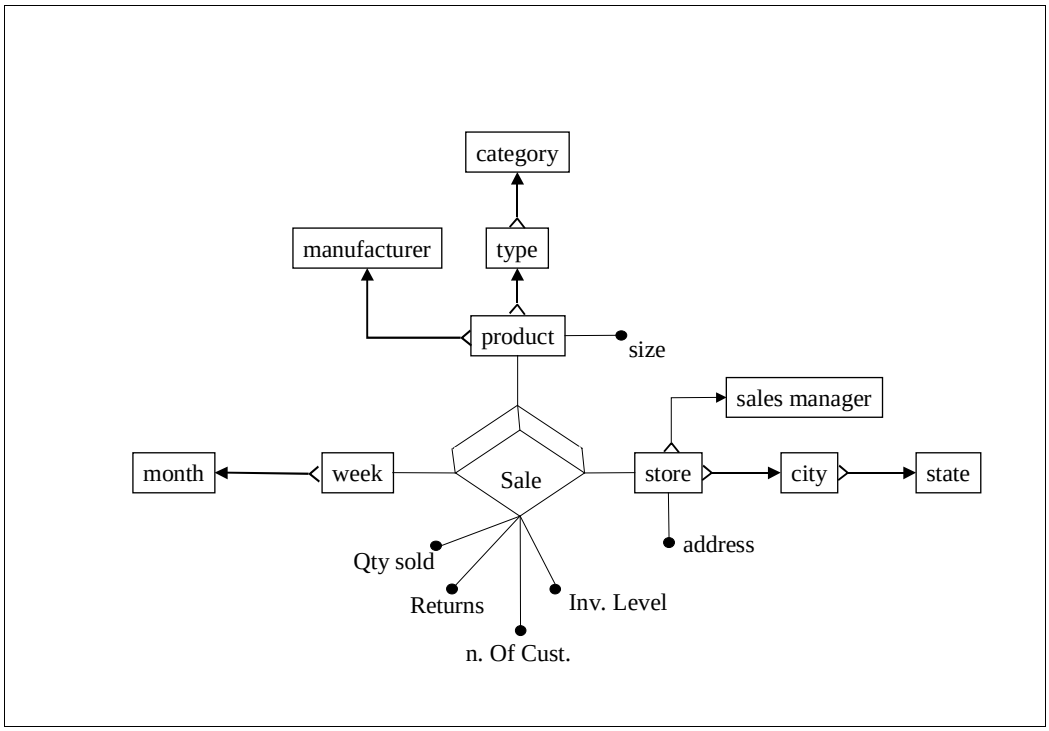

Fig. 8. The Sales Example.

The measures are attributes of a fact relationship set. Description attributes are attributes of dimension levels.

3.4.3. Conclusions.

The model has an advantage, with respect to the Golfarelli proposal: Is based on a standard model, in syntax and semantic. Therefore, the model should be friendly to persons that know E/R. Moreover, the experience in (semi) automatic treatment of E/R may be reused.

The drill-across paths rise in a natural way when two or more fact relationship sets are connected to levels in the same hierarchy.

However, there is no support for generic dimensionality and has the same drawbacks as E/R like no (wide accepted) modular representation.

Is not clear in the paper how the dimension level elements can be identified but an identifier for each one can be supposed.

The model of Cabibbo and Torlone can be saw as a more generic semantic for this model. To do this, it can be identified fact relationship set with fact tables dimension levels with the same name in the Cabibbo's model.

3.5. Franconi – Sattler. (DWQ)

"Foundations of Data Warehouse Quality" (DWQ) was a long-term research project funded by the European Commission under the ESPRIT program. The interest of the project goes from conceptual to physical aspects of Data Warehouse.

The Data Warehouse Conceptual Data Model (DWCDM) is its proposal for conceptual modeling.

The model has two languages:

- A graphical language based on E/R diagrams.
- A basic language based on Description Logics.

In [Fra99a], the authors present an overview of both languages and a sketch from a method to translate the graphical language in the basic language.

In [Fra99] the authors expose the fundamentals of they proposal and give a good description of the basic language.

The rest of the section is an attempt to explain this model based on these papers. It must be taking in account that there is no description of the graphical language and the basic languages in both papers are softly different.

3.5.1. Fundamentals.

All the assumptions made previously are valid in this model.

The first difference is the vision of a dimension: A dimension is an attribute domain potentially structured with multiple hierarchies. Therefore, *there is no structure to represent explicitly a dimension.* The dimensions are represented as relationship sets and its levels are represented as entity sets. Which relationship set is a dimension can be inferred from the aggregation notation.

The model is oriented to express the structure of aggregations. This idea involves abstract properties of aggregations, relationships between the aggregation and their components. However, the description of an aggregation not includes a specification of how its attribute values must be computed in function of the components attributes values.

When an expression like *average(age)* or *min(income)* appears in a diagram, it must be understood as an attribute name.

3.5.2. Graphical Language.

The graphical language is an extension of E/R diagrams[10], adapted to represent explicitly the aggregations.

An example proposed in [Fra99a] talks about a telephone company. In the Fig. 9 a diagram of the base data for a data warehouse for this company can be saw.

Which is the difference between this diagram and an E/R from the operational data? The logic of the construction is different. The Fig. 10 presents a diagram sketch that, possibly, it is closer to the operational data.

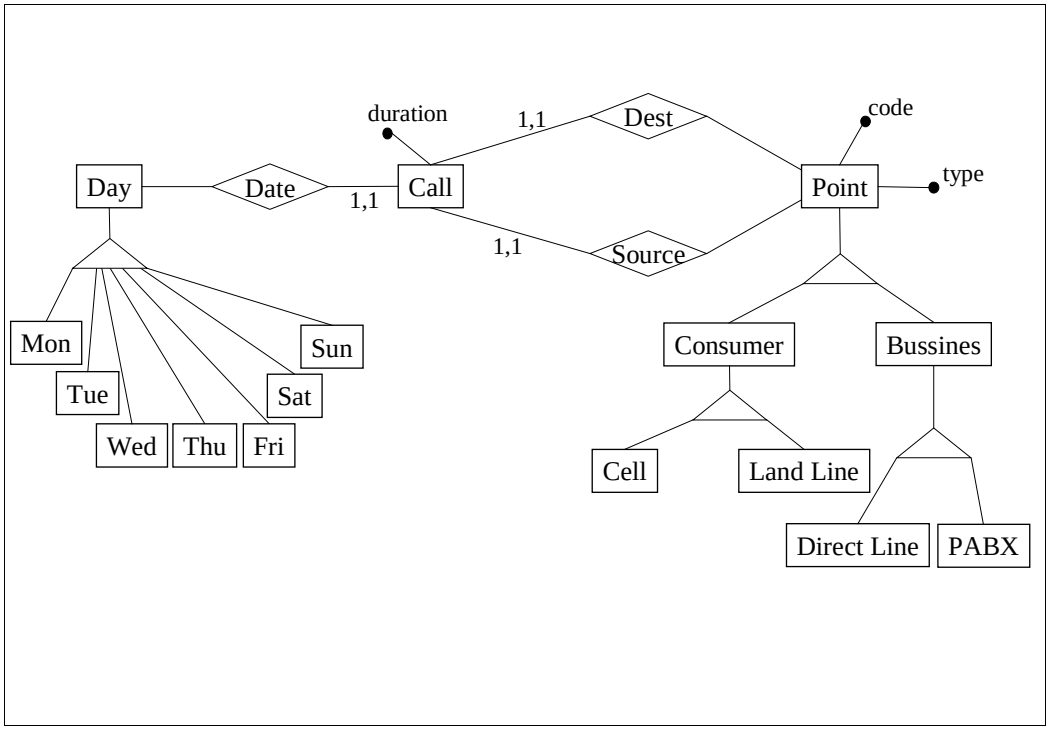

Fig. 9. Conceptual Data Warehouse Schema for base data.

[10] The syntax of the E/R diagrams was modified from the original papers to a more standard notation.

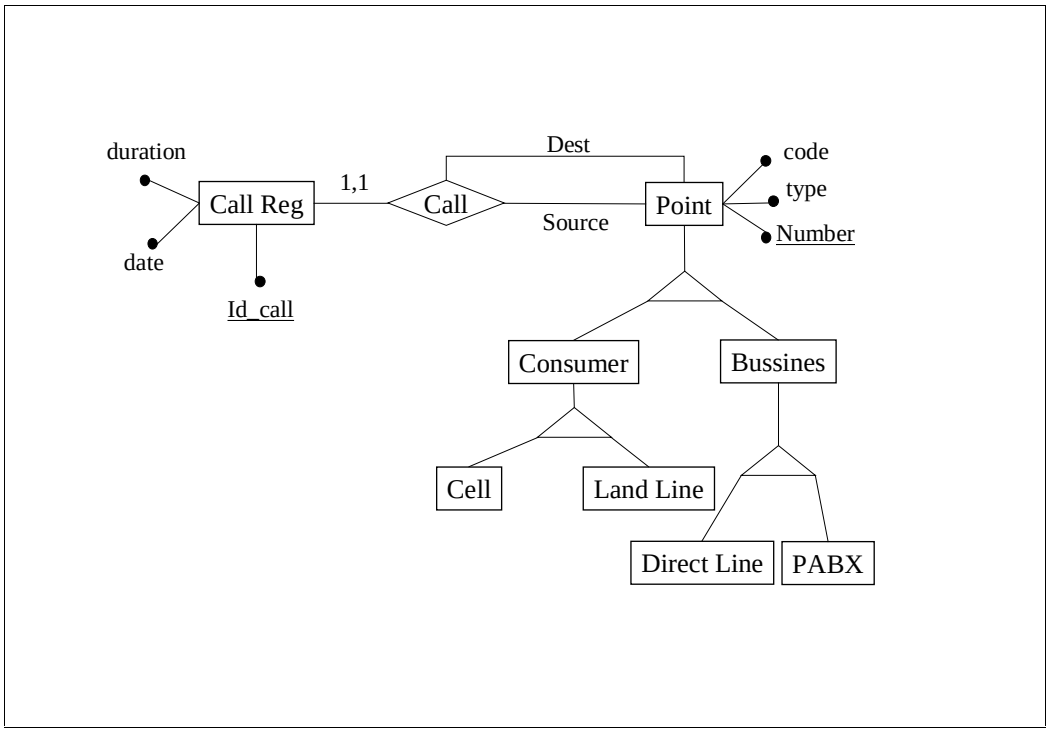

Fig. 10. A possible schema of operational data for the previous DW Schema.

There are at least two reports required by the users: one presents the average of duration by day and point type (cell, land line, etc.) and the other presents the average of duration by weekday and customer type (consumer or Business).

Average(Duration)	date			
Source (point type)	01/01/2000	01/03/2000	01/05/2000	01/07/2000
Direct Line	62,71483396	93,13603339	75,93415712	81,44172689
PABX	119,5707099	15,9396764	67,68244536	...

Fig. 11. Duration average by source point type and date

Average(Duration)	week day				
Source (type of customer)	1	2	3	4	5
Bussines	84,24889464	50,73514859	11,78495917	103,6093389	64,34583368
Consumer	71,40934886	61,32803122	115,6543695	60,18969521	...

Fig. 12. Duration average by source type of customer and web day (1=sunday).

In the Fig. 13 can be saw the conceptual data model for the first cube.

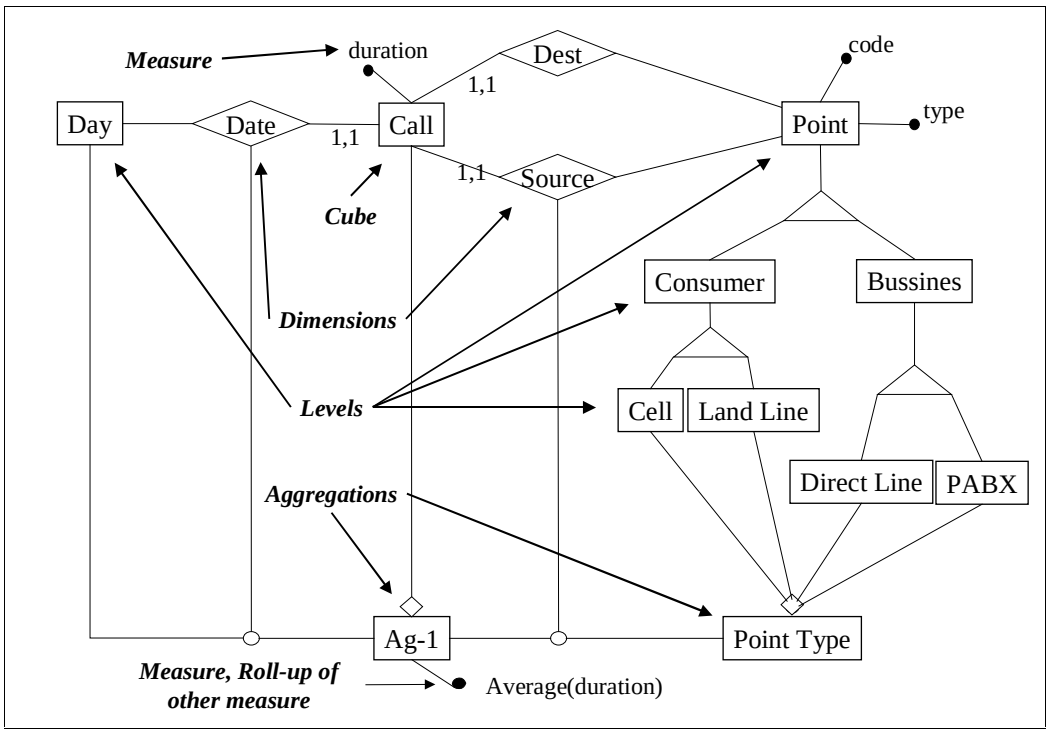

Fig. 13. Conceptual model for first cube.

The aggregation *Ag-1* aggregates *calls* depending on dimensions *Date* and *Source* at levels *Day* and *Point Type*. This last level is an aggregation of all point types. The diamond symbol can be read as a relationship set called *aggregates* with distinguished roles.

The semantic of this graphical model is defined in term of legal states of the multidimensional database. So, to interpret the diagrams, the authors choose the logical model proposed by Cabibbo and Torlone in [Cab98]. An instance of this logical model expresses a legal state for the multidimensional database. If the model is inconsistent then, there is no legal state to any database.

3.5.3. The Basic Language.

The graphical language can be mapped to this basic language. This language is based on Description Logics.

In order to simplify the reading, in the next section a minimal background on the theme will be exposed.

Next, some of the language will be presented.

3.5.3.1. Description Logics.

Description Logics are a family of formalisms for knowledge representation, which are based in the representation of Concepts and Roles.

A concept can be understood as a description for a set of elements or as a unary predicate.

A role can be understood as a description for a binary relation or as a binary predicate.

A description logic has a name according with its features. The Description Logic used by the authors is named ALCFI because is the basic (ALC) with “features” and “inverse roles”.

The syntax used in [Fra99a] is presented in the Fig. 14. In this syntax, C and D are concepts, R and S are roles and f and *g* are features.

A feature is a functional role.

C,D	→	A |	(concept name)
		⊤ |	(top)
		⊥ |	(bottom)
		¬C |	(complement)
		C⊓D |	(conjunction)
		C⊔D |	(disjunction)
		∀R.C |	(universal quantifier)
		∃R.C |	(existencial quantifier)
		f↑ |	(undefinedness)
		f:C |	(selection)
R,S	→	P |	(role name)
		f |	(feature)
		R^{-1} |	(inverse role)
		$R\|_C$ |	(restriction)
		R ° S	(role chain, composition)
f,*g*	→	p |	(feature name)
		f ° *g*	(feature chain, composition)

Fig. 14. Syntax for the ALCFI Description Logic.

$\top^{\Im}$	=	$\Delta^{\Im}$
$\bot^{\Im}$	=	$\varnothing$
$(\neg C)^{\Im}$	=	$\Delta^{\Im} \setminus C^{\Im}$
$(C \sqcap D)^{\Im}$	=	$C^{\Im} \cap D^{\Im}$
$(C \sqcup D)^{\Im}$	=	$C^{\Im} \cup D^{\Im}$
$(\forall R.C)^{\Im}$	=	$\{i \in \Delta^{\Im} \mid \forall j.(i,j) \in R^{\Im} \Rightarrow j \in C^{\Im}\}$
$(\exists R.C)^{\Im}$	=	$\{i \in \Delta^{\Im} \mid \exists j.(i,j) \in R^{\Im} \wedge j \in C^{\Im}\}$
$(f\uparrow)^{\Im}$	=	$\Delta^{\Im} \setminus dom(f^{\Im})$
$(f{:}C)^{\Im}$	=	$\{i \in dom(f^{\Im}) \mid f^{\Im}(i) \in C^{\Im}\}$
$(R^{-1})^{\Im}$	=	$\{(i,j) \in \Delta^{\Im} \times \Delta^{\Im} \mid (j,i) \in R^{\Im}\}$
$(R \circ S)^{\Im}$	=	$R^{\Im} \circ S^{\Im}$

Fig. 15. Extensional Semantic for ALCFI Description Logic

The semantic of Description Logic is defined by means of an interpretation $\Im$. An interpretation $\Im$ is a pair $(\Delta^{\Im},\cdot^{\Im})$ where $\Delta^{\Im}$ is a set of elements (domain of $\Im$) and $\cdot^{\Im}$ is a function which maps each concept in a subset of $\Delta^{\Im}$ and each role in a subset of $\Delta^{\Im} \times \Delta^{\Im}$.

The semantic of ALCFI can be seen in Fig. 15.

For a better comprehension of the ALCFI, its equivalence with First Order Logic is exposed in Fig. 16. A concept C in Description Logic is equivalent to an open formula $F_C(y)$ with one free variable *y* and a Role R is equivalent to an open formula $F_R(y,x)$ with two free variables *y* and *x* ([Art99]).

$\top^{\Im}$	~	**True**
$\bot^{\Im}$	~	**False**
$(\neg C)^{\Im}$	~	$\neg F_C(y)$
$(C \sqcap D)^{\Im}$	~	$F_C(y) \wedge F_D(y)$
$(\exists R.C)^{\Im}$	~	$\exists x.(F_R(y,x) \wedge F_C(x))$
$(\forall R.C)^{\Im}$	~	$\forall x.(F_R(y,x) \Rightarrow F_C(x))$
$(f\uparrow)^{\Im}$	~	$\neg\exists x.f(y,x)$
$(f{:}C)^{\Im}$	~	$\exists x.(f(y,x) \wedge F_C(x))$
$(R^{-1})^{\Im}$	~	$F_R(x,y)$
$(R \circ S)$	~	$\exists z.(F_R(y,z) \wedge F_S(z,x))$

Fig. 16. First Order Logic equivalence for ALCFI

3.5.3.2. The Translation of Extended E/R Diagrams.

The translation consist of constructing a knowledge base (or terminology) Σ, from a diagram D, where there is a concept name for each entity, aggregation, relationship or domain and there is a feature name for each relationship role or attribute. For each type of construction in the diagram, there are some rules that add appropriate terminological axioms to the terminology Σ.

As example, the following is part of the knowledge base for the example:

$Date \sqsubseteq what : Call \sqcap when : Day$

$Source \sqsubseteq what : Call \sqcap where : Point$

$Dest \sqsubseteq what : Call \sqcap where : Point$

$Point \sqsubseteq Consumer \sqcup Business$

$Consumer \sqsubseteq Point \sqcap \neg Business$

$Business \sqsubseteq Point \sqcap \neg Consumer$

The first axiom is the definition for the concept *Date*, which rises from the relationship Date in the diagram. The expression *what* in this axiom is a feature (functional role) and denotes the role between the entity Call and the relationship Date. The expression *when* denotes the role between the relationship Date and the entity Day. The expression *what:Call* is a concept that must be understand as the set of elements from the interpretation domain that are related with any Call by the role *what*. In terms of the extensional semantic, this can be thought as:

$(what:Call)^{\mathcal{I}} = \{ i \in dom(what^{\mathcal{I}}) \mid what^{\mathcal{I}}(i) \in Call^{\mathcal{I}} \}$

The expression *when:Day* is similar to the previous. So, the axiom say that the intersection of the concepts *what:Call* and *when:Day* subsumes the concept *Date*. This is considered as the primitive concept definition for the concept *Date*.

The rest of the axioms in the first group are similar to this.

The first axiom in the second group expresses that any instance of *Point* must be (primitive concept definition again) an instance of *Consumer* or a instance of *Business*. The second and third axioms of this group express that *Consumer* and *Business* conforms a partition of *Point*.

In [Hor99] and [Hor99a] an algorithm based on the tableau method that makes decidable the satisfaction of a terminology is proposed. This is used by Franconi and Sattler to decide the model consistency.

3.5.4. Conclusions.

The proposal is the most complete, at least, until today.

The graphical language does not support generic dimensionality and apparently, there is no a clear way to see the paths on dimension hierarchies. The last remark is evident in the Fig. 13. The entity *Point Type* is a level from the dimension *Source* (and *Dest* too) because is involved by the aggregation Ag-1. However, *Point Type* is a level higher in the dimension hierarchies to the levels *Cell, Land Line, Direct Line and PABX*, but independent from *Consumer and Business*.

There is not any language to express general constraints, but it is possible to suppose that these constraints can be expressed in Description Logics.

4. Other Related Works.

The Design Methodologies are a target in itself. However, all multidimensional models support different methodologies, in spite of some of them were designed very close of a particular methodology ([Gol99], [Gol98], [Cab98]).

Summarizability is really a topic for conceptual modeling. Are all roll-up legal in any cube? It is clear that the answer to this question is not. Some of this is explained in [Kim96] as additive measures, but a more clear and general exposition over this is [Len97].

Lehner and other expose some consideration and definitions on Multidimensional Normal Forms in [Leh98]. The authors define that in multidimensional databases the normal forms must be oriented to two targets: ensure summarizability and reduce the sparseness of the data cube.

In the Description Logics appendix, there are many references on the topic.

There is no found any work, which explicitly says something about constraints on multidimensional models. However, it is possible that Description Logics can express some constraints. In fact, in [Fra99a], the language allows specify some cardinality constraints. In [Hor99], the DL language supports more cardinality constraints, but, in anywhere there is an explanation on how express general constraints in DLs. It is possible that in a DL with this feature the satisfaction not be decidable.

5. A Framework for the Comparison of Multidimensional Models.

To compare conceptual multidimensional models, the following topics must be taken in account:

- **Model Type**. If the model is query oriented or logical or conceptual.
- **Relational Independence**. The model must be multidimensional, not an implementation of a multidimensional model in Relational Model or other data model with a lower conceptual level. This problem makes difficult the implementation of the structures in a real multidimensional system.
- **Definition of dimensional hierarchies**. The model must allow the designer to specify some dimensional hierarchies. Each model has different orientations in this topic.
- **Symmetry between measures and dimensions**. This is a condition for OLAP imposed by Codd in its definition in [Cod93]. A few models treat to this topic.
- **Summarizability**. Only one of the studied model has some support for it.

- **Constraint Language**. A data model must allow to define data structures and must give a precise semantic description providing a language for integrity constraints.

Many other topics can be taken in consideration but these ones seems to concentrate in Conceptual Modeling or semantic of the multidimensional modeling.

Model	Model type	Relational Independence	Definition of dim. hierarchies	Symmetry between Dim. And Measures	Summarizability	Constraint Language
Li	query	no	relations	no	No	no
Agrawal	query	yes	by operations	by operations	No	no
Gyssens	query	no	relations	by operations	No	no explicitly
Cabbibo	logical	yes	structural	by operations	No	no explicitly
Lehner	logical	yes	structural	no	Yes	no
Sapia	conceptual	yes (ER/extension)	structural	no	No	no explicitly
Golfarelli	conceptual	yes	structural	no	Yes	no explicitly
Franconi	conceptual	yes (ER/extension)	structural (but no explicitly)	no	No	no explicitly

Fig. 17. A comparison between the multidimensional models.

In the Fig. 17, in the column *constraint language*, the value *no explicitly* expresses that in spite of there is not a definition of a constraint language in the model, there are elements to think that may be possible a clear definition. When the model has a calculus as a query language, it can be used to express some constraints with a high expressiveness. For Franconi's model, some constraints may be expressed in Description Logics or similar.

In the column *definition of dimensional hierarchies* the Franconi's model has a remark: is structural but not in an explicit way. This remark appears because the model has no explicit structure for the dimensional hierarchies but it can be read from the diagrams as ISA hierarchies and simple aggregations.

6. Conclusions.

The most important multidimensional models were briefly described. There are only three conceptual models. However, all of the models have worth noting properties.

Li ([Li96]) and Gyssens ([Gys97]) models are valuable as Star Schema formalizations.

Agrawal's model ([Agr97]) introduces the notion of Boolean cube, which can be used to manage the symmetry between dimensions and measures.

Cabibbo´s model ([Cab97] , [Cab98]) has a clear and complete formalization of basic multidimensional structures and its calculus introduce a clear way to consult these structures.

Lehner's model ([Leh98a]) has a formal characterization of multidimensional domains (instances) introducing clear definitions of data contexts. In another paper ([Leh98]), there is a definition of Normal Forms over this model.

Sapia´s model ([Sap99a]) has a clear graphical notation based on ER diagrams.

Golfarelli´s model ([Gol99], [Gol98]) has a definition of a method to express drill-across, which seems to be adequate for multidimensional schema integration. Also has a general specification for summarizability properties.

Franconi´s model ([Fra99], [Fra99a]) is based in Description Logics, which allows the implementation of decision procedures over the consistency of the model. Moreover, its graphical language extends ER diagrams with multidimensional and simple aggregations.

A lack in all models is an explicit way to express general constraints. This feature can help the designer with the specification accuracy.

Bibliography.

[Agr97] Agrawal, R. Gupta, A. Sarawagi, S.:*"Modelling Multidimensional Databases"*, ICDE, 1997.

[Art99] Artale, A. Franconi, E:*"Introducing Temporal Descriptions Logics."*, ???? DWQ Internet Site, 1999.

[Cab97] Cabibbo,L. Torlone, R.:*"Querying Multidimensional Databases"*, SEDB, 1997.

[Cab98] Cabibbo, L. Torlone, R.:*"A Logical Approach to Multidimensional Databases"*, ", Sixt Int. Conference on Extending Database Technology (EDBT '98), Springer-Verlag, 1998.

[Cab99] Cabibbo, L. Torlone, R.:*"Data Independence in OLAP Systems"*, Dipartimento de Informatica e Automazione, Università di Roma Tre, 1999.

[Cal98] Calvanese, D. Lenzerini, M. Nardi, D.:*"Description Logics for Conceptual Data Modeling."*, Logics for Database and Information Systems., J. Chomicki and G. Saake eds., 1998.

[Car97] Carpani, F. Goyoaga, J. Fernandez, L.:*"Modelos Multidimensionales"*, IV Jornadas de Informática e Investigación Operativa. Instituto de Computación de la Facultad de Ingeniería. Monteviedo, Uruguay., 1997.

[Cod93] Codd, E.F. Codd, S.B. Salley, C.T.:*"Providing OLAP to user-analysts. An IT mandate."*, Technical Report., E.F. Codd and Associates., 1993.

[Fra99] Franconi, E. Sattler, U.*"A Data Warehouse Conceptual Data Model for Multidimensional Aggregation: A Preliminary Report."*, Technical Report., DWQ, 1999.

[Fra99a] Franconi, E. Sattler, U.:*"A Data Warehouse Conceptual Data Model for Multidimensional Aggregation."*, Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99), S. Gatziu, M.Jeusfeld, M.Staudt, Y. Vassiliou, Eds., Jun. 1999.

[Gol98] Golfarelli, M. Rizzi, S.:*"A Methodological Framework for Data Warehouse Design"*, First International Workshop on Data Warehousing and OLAP (DOLAP), ACM, Nov. 1998.

[Gol98a] Golfarelli, M. Maio, D. Rizzi, S.:*"Conceptual Design of Data Warehouses from E/R Schemes."*, International Conference on Systems Science, Hawaii, IEEE, Ene. 1998.

[Gol99] Golfarelli, M. Rizzi, S.:*"Designing the Data Warehouse: Key Steps and Crucial Issues."*, Journal of Computer Science and Information Management. Vol. 2. N. 3, Maximilian Press Publisher, 1999.

[Gru99] Grumback, S. Rafanelli, M. Tininini, L.:*"Querying Agregate Data"*, Proceedings Conference on Principles of Database Systems (PODS '99), Philadelphia, ACM, 1999.

[Gys97] Gyssens, M. Lakshmanan, L.:*"A Foundation for Multi-Dimensional Databases",* VLDB, Athens, Greece, 1997.

[Hor99] Horrocks, I. Sattler, U. Tobies, S.:*"Practical Reasoning for Expressive Description Logics.",* DWQ Internet Site., 1999.

[Hor99a] Horrocks, I. Sattler, U. Tobies, S.:*"Practical Reasoning for Description Logics with Functional Restrictions, Inverse Roles and Transitive Roles, and Role Hierarchies.",* DWQ Internet Site, 1999.

[Jar99] Jarke, M. Lenzerini, M. Vassiliou, Y. Vassiliadis, P.:*"Fundamentals of Data Warehouses",* , Springer-Verlag, 1999.

[Ken96a] Kenan Technologies:*"An Introduction to Multidimensional Databases.",* White Paper, Kenan Technologies, 1996.

[Kim96] Kimball, Raplh.:*"The Datawarehouse Toolkit.",* Book., John Wiley & Son, Inc., 1996.

[Leh98] Lehner, W. Albrecht, J. Wedekind, H.:*"Normal Forms for Multidimensional Databases.",* , 1998.

[Leh98a] Lehner, W.:*"Modeling Large Scale Olap Scenarios.",* Proc. EDBT '98. Valencia. España., http://www6.informatik.uni-erlangen.de/dept/staff/lehner-publications.html, 1998.

[Len97] Lenz, H. Shoshani, A.:*"Summarizability in OLAP and Statistical Databases.",* 9th Int. Conf. on Statistical and Scientific Databases , 1997.

[Li96] Li, C. Wang, X.S.:*"A Data Model for Supporting On-Line Analytical Processing",* Conf. on Information and Knowledg Management, Nov. 1996.

[Lou92] Loucopoulus, P. Zicari, R.:*"Conceptual Modeling Databases and CASE",* , John Wiley & Sons, Inc., 1992.

[Man99] Mangisengi, O. Tjoa, A M.:*"A Multidimensional Modeling Approach for OLAP within the Framework of the Relational Model based on Quotient Relations",* Proceedings Conference on Principles of Database Systems. (PODS ´99) Philadelphia, ACM, 1999.

[Sap99] Sapia, C. Blaschka, M. Höfling, G.:*"An Overview of Multidimensinal Data Models for OLAP",* Technical Report., http://www.forwiss.de/~system42/publications, Feb. 1999.

[Sap99a] Sapia, C. Blaschka, M. Höfling, G. Dinter, B.:*"Extending the E/R Model for the Multidimensional Paradigm.",* Advances in Database Technologies. LNCS Vol 1552, Springer-Verlag. 1999.

[Sap99b] Sapia, C. Blaschka, M. Höfling, G.:*"GraMMi: Using a Standard Repository Management System to Build a Generic Graphical Modeling Tool",* ???, ???, +Ju. 1999.

[Tho97] Thomsen, E.:*"OLAP Solutions. Building Multidimensional Information.",* Book, John Wiley & Sons, Inc. , 1997.

ANEXO II.

Constantes en el Lenguaje de Restricciones de CMDM.

1. Introducción.

En el lenguaje de restricciones existen dos tipos de constantes:

- Constantes predefinidas.
- Constantes generadas a partir del esquema.

Las constantes predefinidas son las mismas para todos los esquemas y representan literales o funciones predefinidas.

Las constantes generadas a partir del esquema, representan los elementos del esquema o los elementos de la instancia del esquema.

En las siguientes secciones se presentan las diferentes constantes y sus tipos.

2. Constantes Generadas a Partir del Esquema.

2.1. Introducción.

Las restricciones de integridad permiten expresar condiciones sobre los diferentes elementos del esquema, por lo que se hace imprescindible tener un mecanismo interno al lenguaje de restricciones de integridad que permita referenciar dichos elementos.

Para solucionar esto, para cada tipo de elemento del esquema se genera una constante de determinado tipo que representa a la instancia de ese elemento.

Además, se cuenta con una función específica que permite acceder al esquema de ese elemento.

En la sección 4.4.4.1 se presenta un resumen de los tipos que deben tener estas constantes.

2.2. Niveles.

Para cada nivel, se debe generar una constante que representa a su instancia. Por lo tanto, estas constantes tiene tipo ***Set(i)***.

Por otro lado, también es necesario introducir las constantes necesarias para representar los elementos de los tipos de los niveles.

De esta forma, en el caso de estudio, se definirían las siguientes constantes para el nivel *Vendedor*.

Constante	Tipo	Significado
Vendedor	*Set(i)*	Instancia del nivel.
Cedula	*i*	Atributo *Cédula* del tipo del nivel
Apellido	*i*	Atributo *Apellido* del tipo del nivel.
Nombre	*i*	Atributo *Nombre* del tipo del nivel.
Edad	*i*	Atributo *Edad* del tipo del nivel.
Sexo	*i*	Atributo *Sexo* del tipo del Nivel
Ant	*i*	Atributo *Antigüedad* del tipo del Nivel
Tel	*i*	Atributo *Teléfono* del tipo del nivel

Fig. 1. Constantes generadas a partir del nivel Vendedor del caso de estudio.

Se asume que el tipo del nivel es un producto cartesiano. También se asume que existe una función ***select:i*→α** tal que aplicada a un nivel y a una constante, representa el valor asociado en la instancia. En los ejemplos, esa función se representa en forma infija con el carácter "**.**".

2.3. Dimensiones.

Para cada dimensión, se genera una constante que representa la instancia de la dimensión.

La instancia de una dimensión es una pareja formada por una función que mapea los niveles en su instancia y una familia de funciones que mapean un nivel en otro superior (Definición 7. y Definición 15.). Por esto, el tipo de las constantes que representan las instancias de dimensiones es el siguiente:

[Set(i)→ Set(i),[Set(i),Set(i)]→(Set(i)→Set(i))]

2.4. Relaciones Dimensionales.

Para cada relación dimensional, también se necesita una constante que represente su instancia. El tipo de esa constante debe ser ***Set(*α*)*** en donde α es el tipo de los cubos.

Un cubo, es una función de las coordenadas en los booleanos, por lo que el tipo α es β→***i***, donde β es el tipo de las coordenadas.

Una coordenada es una tupla, o sea, una función del niveles en los valores de sus instancias. Para poder utilizar directamente las constantes definidas, se asume que la función, va de la instancia del nivel en los booleanos. Por esto, el tipo β es ***Set(i)→i***.

De esta forma, el tipo de los cubos es ***(Set(i)→i)→i*** , por lo que el tipo de las relaciones dimensionales es ***Set((Set(i)→i)→i)***. Cualquier constante que represente un cubo deberá tener este tipo.

3. Constantes Predefinidas.

3.1. Funciones de Acceso.

Para poder expresar condiciones sobre las partes de un elemento del esquema, es necesario tener un mecanismo para representar a esa parte de ese elemento del esquema. Por ejemplo, dada la dimensión *Vendedores* , para poder expresar una condición sobre los niveles que participan en esa dimensión, se utiliza la expresión *DimLevels*.

Dado un elemento del esquema, existen funciones que permiten recuperar las distintas componentes de ese elemento. En las siguientes secciones se presentan las diferentes constantes definidas con este fin, de acuerdo a la estructura sobre la que trabajan.

3.1.1. Niveles.

Sobre los niveles, se necesitan funciones que representan cada uno de los elementos del esquema de cada nivel.

Estas funciones son las siguientes:

- ***Level_name:Set(i)→i***. Devuelve el nombre del nivel.
- ***Level_typeS:Set(i)→i***. Devuelve el tipo del nivel, como un string.

3.1.2. Dimensiones.

Para las dimensiones, además de las funciones que devuelven elementos del esquema, se necesitan funciones que devuelvan las componentes de la instancia.

Para el esquema, se necesitan las siguientes funciones:

- ***Levels: [Set(i)→Set(i),[Set(i),Set(i)]→(i→i)]→Set(Set(i))***. Devuelve el conjunto de niveles de la dimensión.
- ***PO: [Set(i)→Set(i),[Set(i),Set(i)]→(i→i)]→[Set(i),Set(i)]***. Devuelve el conjunto de parejas de niveles que conforman la jerarquía de la dimensión.

Para la instancia, también se necesitan dos funciones:

- ***IL:[Set(i)→Set(i),[Set(i),Set(i)]→(i→i)]→(Set(i)→Set(Set(i)))***. Devuelve la primer componente de la instancia de la dimension.
- ***DrillUp:[Set(i)→Set(i),[Set(i),Set(i)]→(i→i)]→([Set(i),Set(i)]→(i→i))***. Devuelve la función de DrilUp correspondiente a la dimensión.

3.1.3. Relaciones Dimensionales.

El esquema es sólo un conjunto de niveles, por lo que la única función para el esquema es la siguiente:

RD_Levels: Set((Set(i)→i)→i)→Set(Set(i)).

Con respecto a las instancias, es sólo un conjunto de cubos, por lo que no necesita de ninguna función en particular. Sin embargo, hay al menos una función sobre los cubos:

- ***Dom: ((Set(i)→i)→i)→Set(Set(i))***. Devuelve el conjunto de niveles que participan en el cubo.

3.2. Otras Funciones Predefinidas.

3.2.1. Funciones Aritméticas.

Se asume que cualquier símbolo aritmético debe ser entendido como la propia función que representa el símbolo.

3.2.2. Funciones de Agregación.

Una función de agregación es cualquiera con tipo ***Set(*α*)→*β donde α y β son tipos cualesquiera.

Las funciones más usadas serán las que mapean subconjuntos de instancias de niveles en otros elementos de Δ, por lo que tendrán tipo: ***Set(i)→i***. Funciones con este tipo son Sum y Avg.

Sin embargo, no se ha encontrado por el momento ningún argumento fuerte para prohibir funciones, por ejemplo, de un conjunto de funciones en una sola función cuyo tipo podría ser ***Set(i→i)→(i→i)***.

La función ***Count:Set(*α*)→i***, devuelve el cardinal del conjunto que recibe como parámetro.