UNIVERSIDAD POLITÉCNICA DE VALENCIA

Departamento de Sistemas Informáticos y Computación

Tesis doctoral

Metodología para el Diseño Conceptual de Almacenes de Datos

Presentada por:

Leopoldo Zenaido Zepeda Sánchez

Dirigido por:

Dra. Matilde Celma Giménez

Valencia, España, junio de 2008

.

Resumen

Desde la introducción del modelo de datos multidimensional como formalismo de modelado para Almacenes de Datos, han aparecido en la literatura sobre el tema, distintas propuestas metodológicas para capturar la estructura del almacén de datos a nivel conceptual. Las soluciones siguen diferentes aproximaciones al diseño: el análisis de los requisitos de usuario, el análisis del esquema de la base de datos operacional o una combinación de ambas aproximaciones (técnicas compuestas).

Por otro lado, Model Driven Architecture (MDA) irrumpe como un nuevo estándar para el desarrollo de sistemas, dirigido por modelos.

Esta tesis se enmarca en el área del diseño de almacenes de datos, en ella se propone una nueva metodología para el diseño conceptual de almacenes de datos en el contexto de la propuesta de MDA. La metodología consta de tres fases.

En la primera fase se analiza el esquema conceptual de la base de datos operacional, generando de forma semiautomática los posibles esquemas multidimensionales candidatos para el almacén de datos. La solución a esta fase, se ha abordado en el contexto de MDA para esto, se ha definido un conjunto de reglas de transformación entre el metamodelo Entidad Relación (ER) y el metamodelo multidimensional (OLAP).

En la segunda fase se analizan los requisitos de los usuarios, recogidos por medio de entrevistas. El objetivo de esta fase es obtener información acerca de las necesidades de análisis de los futuros usuarios del sistema. Como base para esta fase, se ha adaptado un método de elicitación de requisitos basado en metas.

La tercera fase, contrasta la información obtenida en la segunda fase, con los esquemas multidimensionales candidatos generados en la primera fase obteniendo así, la solución (soportada por las bases datos operacionales) que capture mejor los requisitos de usuario.

Abstract

Since the introduction of the multidimensional data model as modeling formalism for Data Warehouses (DWs) design, several techniques have been proposed to capture multidimensional properties at the conceptual level.

These techniques have mostly focused on different specific fields such as: user requirements, operational database schema analysis or mixed approaches (is not only initialized from user's requirements, but also from the operational database schema).

On the other hand, MDA is a new standard to develop software by transforming models, for this MDA proposes three viewpoints: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM).

This thesis is developed in the context of conceptual design of DWs with MDA (a methodology for DW conceptual design). This method is employed within a mixed approach and consists of three phases.

The first phase, is devoted to examining the ER schema of the operational database, finding out candidate multidimensional schemas for the DW, this is achieved through a set of transformation rules between the Entity Relationship (ER) PIM and the On-Line Analytical Processing (OLAP) PIM.

In the second phase, user requirements are collected. The purpose of this phase is to gather information from business analysts and/or managers about the company goals and needs. For the basis of our elicitation method, we adopt a goal model.

The final phase integrates these two viewpoints, and thus generates a feasible solution (supported by the existing data sources) that best reflects the DW user's requirements.

Resum

A partir de la introducció del model de dades multidimensional com formalisme de modelatge per a Magatzems de Dades (MDs), s'han realitzat diferents propostes metodològiques per a capturar l'estructura del MD a nivell conceptual. Les solucions proposades parteixen de diferents aspectes de disseny: els requisits d'usuari, l'anàlisi de l'esquema de la base de dades operacional o una combinació d'ambdós (tècniques mixtes).

Model Driven Architecture (MDA) és un nou estàndard per al desenvolupament de sistemes dirigit per models. MDA proposa tres punts de vista: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM).

Aquesta tesi, s'emmarca en l'àrea del disseny de MDs amb MDA (una metodologia per al disseny conceptual de MDs). Aquest mètode, és emprat amb una metodologia composta i consisteix de tres fases. La primera fase, aquesta dedicada a examinar l'esquema ER de la base de dades operacional, generant els esquemes multidimensionales candidats per al AD. La solució a aquesta fase, s'ha abordat en el context de MDA per a això, hem definit un conjunt de regles de transformació entre el PIM Entitat Relació (ER) i el PIM On-line Analytical Processing (OLAP).

En la segona fase, els requisits d'usuari són recollits per mitjà d'entrevistes. El propòsit de les entrevistes és obtenir informació sobre les necessitats d'anàlisis dels usuaris. Com base per a aquesta fase, vam adaptar un mètode de elicitación de requisits basat en metes.

La tercera fase, contrasta la informació obtinguda en la segona fase, amb els esquemes multidimensional candidats formats en la primera fase generant així, la millor solució (suportada per les bases dades operacionals) que millor reflecteixen els requisits d'usuari.

Tabla de contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

En este capítulo, se presenta el origen de la tecnología de almacenes de datos y sus características principales, así como su estado de desarrollo actual.

A continuación se exponen la motivación y los objetivos de este trabajo de tesis, para finalizar con una descripción de la estructura de este documento.

1.1. Almacenes de datos: origen y características.

El desarrollo en las últimas décadas de la tecnología de bases de datos, ha conducido a una situación en la que las organizaciones disponen de grandes volúmenes de datos con información histórica, almacenados en soporte informático.

Así, una vez satisfecha la necesidad de disponer de un sistema de información para la gestión, las organizaciones exigen más prestaciones a sus sistemas, y contemplan la posibilidad de poder extraer conocimiento de la información histórica almacenada, conocimiento que les permita analizar la organización, prever su evolución y tomar decisiones estratégicas para el futuro. Este es el punto de partida de la tecnología de almacenes de datos (ADs).

En la figura 1.1, se muestra la estructura de un sistema de AD, en ella se observa como el almacén integra datos procedentes de distintas fuentes de datos, tanto internas como externas a la organización.

Esta integración es el resultado de un proceso (Extracción, Transformación y Carga (ETL)[1]), en el que los datos son preparados de la forma más adecuada para facilitar el análisis. Los usuarios del almacén realizan este análisis por medio de herramientas OLAP (On Line Analitycal Procesing) y de Minería de Datos (Data Mining), herramientas que permiten explorar los datos almacenados y sacar conocimiento a partir de ellos.

[1] Del inglés *Extraction, Transformation and Load (ETL).*

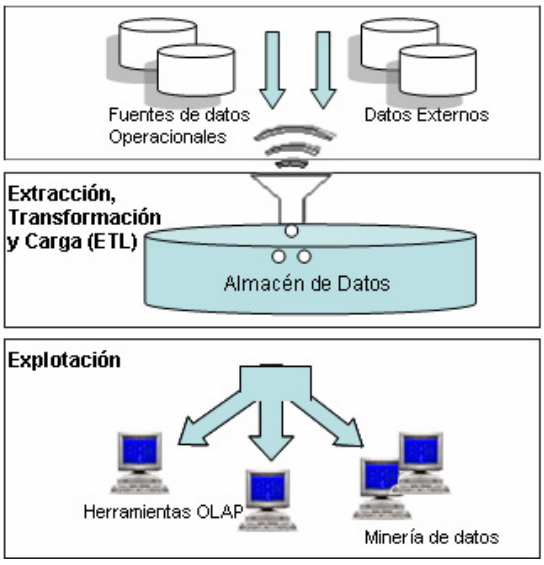


Figura 1.1. Sistema de almacén de datos

Las características especiales de volumen y explotación de los datos en este tipo de sistemas, así como el especial objetivo de uso, han abierto nuevas direcciones de estudio e investigación en el área de Bases de Datos: nuevas estructuras de almacenamiento, optimización de consultas, técnicas de indexación, herramientas de consulta, etc.

Aún así, se observa que el área ha tardado en mostrar interés por los aspectos relacionados con el diseño de este tipo de sistemas, debido posiblemente a que esta tecnología se desarrolló originalmente en el ámbito industrial, preocupado más por problemas tecnológicos que por los aspectos metodológicos de análisis y diseño.

1.2. Motivación y objetivos de la tesis.

En los últimos años se han propuesto distintas metodologías para el diseño de ADs, aunque ninguna de ellas ha sido aceptada plenamente.

En algunos casos, las metodologías son extensiones de las metodologías clásicas para bases de datos, en otros casos se ha adoptado un enfoque completamente nuevo.

Intentando analizar el trabajo hecho hasta el momento en el área del diseño, las propuestas metodológicas pueden clasificarse en tres grupos: metodologías dirigidas por datos, metodologías dirigidas por procesos y metodologías compuestas (datos-procesos).

El objetivo de las metodologías dirigidas por datos es obtener el esquema conceptual del AD a partir de la descripción de las bases de datos operacionales de la organización, por el contrario, las metodologías dirigidas por procesos derivan el esquema conceptual del AD a partir de los requisitos de usuario. Finalmente, las metodologías compuestas realizan una combinación de las dos aproximaciones anteriores, es decir, consideran los requisitos de usuario así como la descripción de la base de datos operacional.

Aunque, como se ha dicho, no existe una metodología que esté ampliamente aceptada, existe un punto de encuentro entre todas ellas, que es la adopción de un modelo de datos multidimensional, que mas que un modelo de datos es una filosofía o estilo de modelado que puede ser aplicado utilizando modelos de datos clásicos (ER, UML, …).

Por otro lado, un enfoque que ha irrumpido con fuerza, en los últimos años, en el área del diseño, ha sido el “diseño dirigido por modelos”. En esta dirección, la OMG ha propuesto el estándar Model Driven Architecture (MDA). MDA consiste en un conjunto de estándares que permiten la creación, implementación, evolución y desarrollo de sistemas por medio de la transformación de modelos [KWB03].

Esta tesis, se enmarca en el área del diseño de ADs. Su objetivo principal es:

"La definición de una metodología de diseño conceptual de ADs, que integre el análisis de la descripción conceptual (esquema Entidad-Relación) de las bases de datos operacionales y el análisis de los requisitos de los usuarios del sistema."

En esencia, la metodología que se propone consta de una estrategia de derivación semiautomática de "esquemas multidimensionales candidatos" a partir del esquema conceptual de la base de datos operacional, empleando para ello un conjunto de reglas de transformación definidas en el marco de MDA. Los esquemas multidimensionales así obtenidos son posteriormente refinados y filtrados basándose en el análisis de los requisitos de usuario.

De acuerdo con esto, los objetivos específicos de este trabajo de tesis son:

1. La definición de un método para identificar el conjunto de esquemas multidimensionales (candidatos) implícitos en el esquema conceptual de la base de datos operacional.

   -definición de un conjunto de reglas de transformación (en el marco de MDA) entre el metamodelo ER y el metamodelo OLAP.

   -implementación de éste conjunto de reglas de transformación que dan soporte a nuestra metodología.

2. La adaptación de un método basado en metas para identificar los requisitos de usuario. Este método integra tres técnicas complementarias para la definición de requisitos.

3. La integración final de estos dos métodos en una metodología única para el diseño conceptual de almacenes de datos.

1.3. Organización del documento de tesis.

La tesis, ha sido organizada en seis capítulos y un anexo. A continuación se describe esta organización:

- **Capítulo 2. Introducción a los ADs.** En este capítulo, se presentan los conceptos básicos sobre ADs, así como los procesos que intervienen en el desarrollo de un AD. En él, se hace también una introducción al modelado multidimensional y sus características principales.
- **Capítulo 3. Introducción a MDA.** En este capítulo, se presenta una breve introducción a Model Driven Architecture (MDA), así como a la tecnología desarrollada alrededor de este estándar.
- **Capítulo 4. El diseño conceptual de ADs: estado del arte.** En este capítulo, se presenta una revisión del estado del arte en el área del diseño conceptual de ADs. Las metodologías que aquí se presentan, son clasificadas en tres grupos de acuerdo a la aproximación que siguen para obtener el esquema conceptual del AD.
- **Capítulo 5. Metodología para el diseño conceptual de almacenes de datos.** En este capítulo, se propone una metodología para el diseño conceptual de ADs. La metodología consta de tres fases: 1) Derivación de esquemas multidimensionales. 2) Análisis de requisitos de

usuario. 3) Integración. La metodología que planteamos se basa en los estándares que integran MDA y en las herramientas que se han desarrollado alrededor de este estándar.

- **Capítulo 6. Conclusiones y trabajos futuros.** En este capítulo, se presentan las conclusiones a las que se han llegado con el desarrollo de esta tesis, se detallan las contribuciones más importantes que se han realizado, las publicaciones derivadas del trabajo de investigación y por último, los trabajos futuros previstos.
- **Anexo A. Programación en QVT.** En éste anexo, se presenta la implementación de las reglas de transformación descritas en el capítulo 5, utilizando el lenguaje QVT.

Capítulo 2

Introducción a los almacenes de datos

En este capítulo se muestran los conceptos básicos sobre almacenes de datos así como los procesos que intervienen en la creación y uso de un sistema de almacén de datos.

En él, se hace también una introducción al modelado multidimensional y sus características principales.

2.1. Introducción.

La definición de almacén de datos, más extendida, es la propuesta por Bill Inmon: "Un almacén de datos (AD) es una colección de datos orientados al dominio, integrados, no volátiles y variables en el tiempo, organizados para dar apoyo al proceso de toma de decisiones" [Inm02].

Esta definición, incluye el objetivo (ayuda a la toma de decisiones) y las principales características (orientados al dominio, integrados, no volátiles y variables en el tiempo). A continuación, se explican con detalle cada una de estas características.

Integrados: En las organizaciones generalmente se utilizan distintos sistemas operacionales, cada uno de ellos con su base de datos diseñada para soportar los procesos específicos de un área del negocio. En la construcción del AD de la organización, datos de todos estos sistemas operacionales deben ser integrados en una única base de datos, este proceso de integración implica costosas tareas de limpieza, transformación y derivación de datos.

Orientados al dominio: En los sistemas operacionales, los datos son estructurados para dar soporte a los procesos básicos del negocio. Así, los mismos datos pueden estar organizados de manera diferente en sistemas operacionales distintos. En el AD los datos se estructuran por "temas de interés" para facilitar su análisis por parte de los usuarios.

Variables en el tiempo: Generalmente, en los sistemas operacionales los datos no tienen una dimensión temporal explícita ya que en estos sistemas se almacena información

actual (último año, último periodo, etc). En un AD sin embargo, el tiempo adquiere un valor importante. Cuando se analizan los datos para descubrir tendencias, es importante conocer "la variación de los datos en el tiempo", por este motivo, en un AD los datos deben estar ligados siempre a un instante específico de tiempo o a un intervalo.

No volátil: En los sistemas operacionales, los datos son almacenados por cortos periodos de tiempo, por ejemplo un año, ya que son de interés para la empresa durante ese periodo. En el análisis de datos, sin embargo es frecuente buscar tendencias del negocio haciendo comparaciones entre los datos de diferentes periodos de tiempo. El AD, existe para ser consultado y no para se modificado. La información es por tanto permanente y la actualización del AD consiste exclusivamente en la incorporación de datos correspondientes al último periodo de tiempo.

2.1.1. Procesos que intervienen en la construcción y uso de un almacén de datos.

Para comprender mejor qué es un sistema de AD, es interesante considerar los procesos que intervienen en su construcción y uso. A continuación se describe cada uno de ellos:

Extracción: Obtención de información de las distintas fuentes de datos operacionales tanto internas como externas a la organización. El principal problema en este proceso, reside en poder acceder a la información que se desea tener almacenada en el AD.

Transformación y carga: El proceso de transformación lo componen una serie de tareas: limpieza, integración, agregación.

Los dos problemas más importantes en la integración son la integración de formato y la integración semántica.

- *Integración de formato*: Se refiere a la unificación de tipos de datos, unidades de medida, codificaciones, etc. Una situación normal en estos entornos es que cada sistema operacional haya sido desarrollado independiente de los otros, dándose situaciones de inconsistencia en el formato y representación de los mismos datos.

- *Integración semántica*: La integración semántica se refiere a la integración de los datos de acuerdo a su significado. Debido a que la información de un AD proviene de diferentes sistemas operacionales diseñados con fines distintos, pueden darse situaciones en las que datos similares tengan significado distinto, o al revés, que datos distintos tengan el mismo significado, pudiéndose plantear problemas en el momento de integrarlos en el AD.

Explotación: Consiste en la consulta y análisis de los datos en el AD. Desde el punto de vista del usuario, el único proceso visible es el de la explotación del AD, aunque la calidad del AD y su éxito radican en los dos procesos anteriores, que durante el desarrollo del AD, consumen la mayor parte de los recursos.

2.1.2. Explotación de un almacén de datos: herramientas OLAP.

Las herramientas OLAP (On-Line Analytical Processing), constituyen una tecnología de software específica para el análisis de datos en un sistema de AD. Aunque las herramientas clásicas de consulta y explotación en bases de datos (generadores de informes) podrían ser utilizadas para este fin, las características de uso y explotación específicas de los sistemas de AD, han favorecido el desarrollo de herramientas orientadas específicamente al análisis de datos.

Una característica común a este tipo de herramientas es la presentación de los datos objeto de análisis en un espacio multidimensional que facilita el análisis desde diferentes puntos de vista.

La idea básica de la perspectiva multidimensional, consiste en presentar al usuario los datos sobre la actividad objeto de análisis en relación con los parámetros o dimensiones que caracterizan la actividad, en un espacio multidimensional.

Consideremos un espacio tridimensional. Un punto cualquiera de este espacio queda determinado por la intersección de tres valores en cada uno de los ejes. Si por ejemplo, representamos en el eje X *productos*, en el eje Y *regiones* y en el eje Z *tiempo en meses*, se podría tener, la siguiente combinación de valores {producto = manzana, región = norte, tiempo = diciembre-2002}, la intersección de estos valores define un punto en el espacio que puede representar cualquier actividad caracterizada por estos

valores (producto, región y tiempo), por ejemplo “las *ventas* de manzanas en la región Norte en diciembre de 2002”.

En esta presentación multidimensional de los datos, en cada eje se representa una dimensión de la actividad objeto de análisis, en la figura 2.1 las dimensiones son *tiempo*, *producto* y *región,* y los puntos del espacio representan la actividad a analizar, en la figura “las v*entas* de un producto en una región en un mes determinado”, descrita esta actividad por un conjunto de datos.

Esta perspectiva multidimensional ha sido utilizada tradicionalmente en el área del análisis de datos, mucho antes de que apareciera la tecnología que ahora nos ocupa, y de que existiese el desarrollo informático que ha favorecido su evolución.

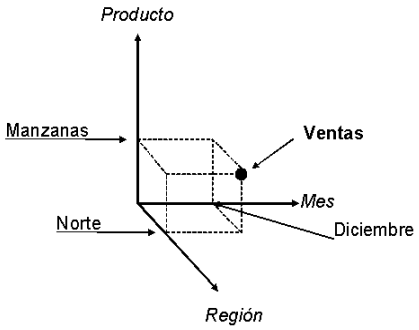


Figura 2.1. Perspectiva multidimensional de los datos

Para explicar de manera más sencilla los conceptos multidimensionales utilizados por la tecnología OLAP, usaremos el ejemplo de la figura 2.2, en el que sólo aparecen dos dimensiones. La tabla representa “las *ventas* de *productos* por *regiones*”. Los productos son: manzanas,

naranjas, peras y toronjas, y se venden en tres regiones: este, oeste, centro.

Producto	Región	Ventas
Manzanas	Este	50
Manzanas	Oeste	60
Manzanas	Centro	100
Peras	Este	40
Peras	Centro	70
Peras	Oeste	80
Naranjas	Este	90
Naranjas	Centro	120
Naranjas	Oeste	140
Toronjas	Este	20
Toronjas	Centro	40

Figura 2.2. Relación de ventas de productos por región

Una representación más clara de esta información puede hacerse en una matriz de dos dimensiones como se muestra en la figura 2.3, donde las dos dimensiones son *producto* y *región*.

REGION / PRODUCTO	Este	Oeste	Centro
Manzana	50	60	100
Peras	40	70	80
Naranjas	90	120	140
Toronjas	20	10	30

VENTAS

Figura 2.3. Representación multidimensional de las ventas

En la terminología OLAP, a los puntos del espacio multidimensional, figura 2.4, se les denomina **celda**, estas celdas contienen información (**medidas**) sobre la actividad que es objeto de análisis.

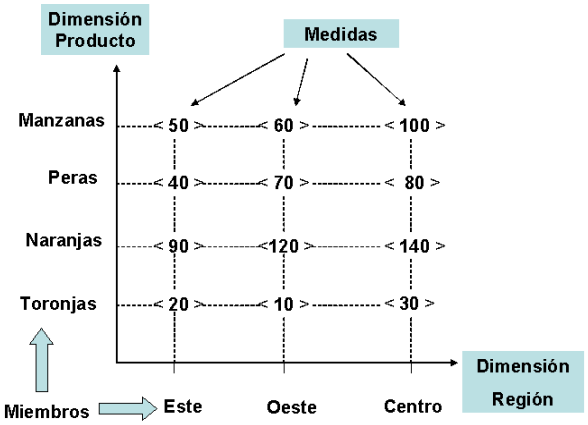


Figura 2.4. Representación multidimensional de los datos

Cuando en un esquema multidimensional participan más de dos dimensiones es común utilizar un cubo de datos para representar los conceptos multidimensionales, en la figura 2.5 se muestra un cubo con tres dimensiones (tiempo, producto

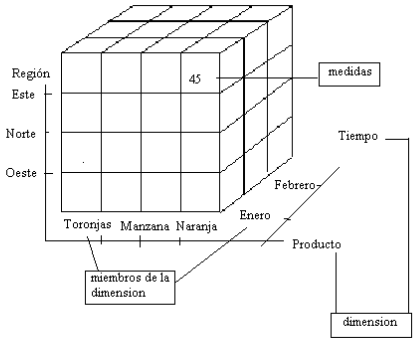


Figura 2.5. Vista multidimensional usando un cubo

y región), en el cubo cada celda contiene una o varias medidas cuyo valor tiene un significado para la intersección

de los valores en las tres dimensiones. (Por ejemplo, el valor 45 se interpreta como el total de naranjas vendidas en la región este en el mes de enero).

Resumiendo, la perspectiva multidimensional consiste en representar la actividad objeto de análisis en un espacio multidimensional. Cada eje del espacio representa una **dimensión** de la actividad y los puntos del espacio (**celdas**) los datos sobre la actividad (**medidas**) para cada combinación de valores.

En esta representación multidimensional de los datos, las dimensiones juegan un papel muy importante ya que representan los puntos de vista del análisis. En la presentación multidimensional, las dimensiones son representadas al nivel de detalle (**gránulo**) al que se desea registrar la actividad en el AD (en el ejemplo "ventas por producto, región y mes"), esto limita significativamente el tipo de análisis que se puede realizar, sólo podremos consultar las ventas por producto, región y mes o cualquier combinación de ellos.

Para enriquecer las posibilidades de análisis, las dimensiones se completan con un conjunto de atributos descriptivos. Estos atributos van a realizar distintas funciones durante el análisis de los datos: permitir establecer condiciones para filtrar los datos, definir criterios de agregación para obtener datos resumidos, etc.

Supongamos, que tenemos almacenada información sobre "ventas de productos, por tienda y mes" (un gránulo inferior al que se ha utilizado en el ejemplo de la figura 2.5), y consideremos la dimensión Tienda.

La descripción de esta dimensión, se puede enriquecer con un conjunto de atributos que describen cada tienda: ciudad en la que se localiza la tienda, estado de la ciudad y región a la que pertenece el estado, además de otros atributos descriptivos como podrían ser características sobre la tienda: tipo de superficie, número de empleados, etc. Así, la dimensión Tienda quedaría descrita como Tienda = {tienda, ciudad, estado, región, superficie, empleados, ...}.

Cuando una dimensión se completa con atributos descriptivos, es usual que estos atributos no sean independientes entre sí, entre ellos suelen aparecer dependencias funcionales que definen jerarquías dentro de la dimensión, estas jerarquías van a desempeñar un papel importante durante el análisis.

En el ejemplo, una jerarquía definida en la dimensión Tienda sería (figura 2.6): tienda → ciudad → estado → región, así para el conjunto de valores representados en el ejemplo, las tiendas 1 y 2 pertenecen a la ciudad Monterrey, las ciudades Monterrey y Apodaca al estado de Nuevo León y los estados Nuevo León y Jalisco a la región Este.

Este enriquecimiento de las dimensiones ofrece nuevas posibilidades de análisis, ahora las consultas pueden ser de de la forma “ventas de productos por regiones realizadas en tiendas de tipo “gran superficie”.

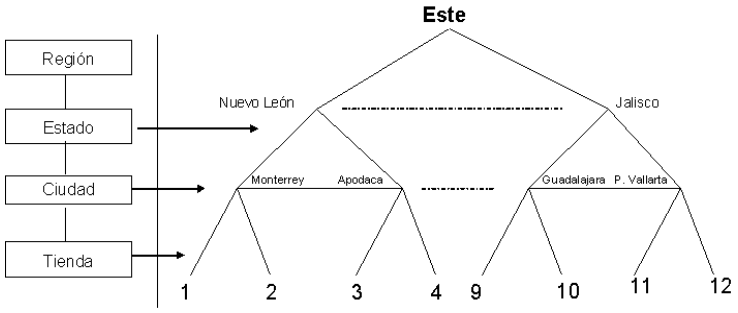


Figura 2.6. Relación jerárquica entre atributos de una dimensión

En esta consulta se observa la función de los atributos descriptivos: el atributo “superficie” nos permite filtrar los datos analizados, restringiéndolos a un subconjunto del almacén de datos; el atributo “región” nos permite obtener información de ventas resumida por regiones a partir de la información de ventas registrada en el almacén por tiendas. Es importante destacar que los datos (medidas) almacenados en el AD son siempre relativos al atributo inferior en la jerarquía, en el ejemplo las ventas de productos son referentes a las tiendas.

La jerarquías definidas van a permitir, además cambiar dinámicamente el nivel de agregación al que se observan los datos de una consulta, pudiendo cambiar en la consulta anterior de región a estado, de región a ciudad, o cualquier salto en la jerarquía definida dentro de la dimensión Tienda, por ejemplo, se podría consultar cuál ha sido el total de ventas de cada producto en cada ciudad (figura 2.7) a partir de una consulta que mostrase las ventas por producto y región (figura 2.8).

Producto \ Ciudad	Monterrey	Apodaca	Morelos	Toluca	Guadalajara	P. Vallarta	Total
Manzana	30	50	50	50	40	20	210
Pera	30	40	40	40	40	30	190
Naranja	40	80	80	60	60	60	350
Toronja	10	10	10	20	5	5	60
Total	110	90	180	170	145	115	810

Figura 2.7. Ventas por producto y ciudad

PRODUCTO \ REGION	Este	Oeste	Centro
Manzana	50	60	100
Peras	40	70	80
Naranjas	90	120	140
Toronjas	20	10	30

Figura 2.8. Ventas por producto y región

Para ello las herramientas de OLAP, introducen un tipo nuevo de operadores, los operadores de DRILL y de ROLL, que permiten cambiar el nivel de agregación de los datos de una consulta, “navegando” a través de las jerarquías.

ROLL-UP: El operador roll-up, permite reducir el nivel de detalle al que se consultan los datos, realizando agregaciones a través de las jerarquías de las dimensiones. Por ejemplo, considere la relación jerárquica de la figura 2.6, la operación roll-up permite cambiar de nivel de Tienda a Ciudad, realizando de nuevo el cálculo para la medida.

DRILL-DOWN: Esta operación, es la inversa de la operación roll-up, es decir permite aumentar el detalle al que se

consultan los datos, al ir a un nivel más bajo dentro de la jerarquía. Considerando de nuevo la jerarquía de la figura 2.6, este operador permite pasar del nivel Región al nivel Ciudad.

Otros operadores incorporados en las herramientas de OLAP son los operadores de SLICE y DICE. Estos operadores permiten reducir el conjunto de datos consultados, por medio de operaciones de proyección y selección de los datos basándose en los atributos de las dimensiones.

La operación de SLICE, consiste en eliminar una dimensión de la consulta activa restringiendo los valores de dicha dimensión a un valor o a un rango de valores.

Por ejemplo, si tenemos una consulta “ventas de productos por región y por mes” y si el dominio para región es {Este, Centro, Oeste}, esta operación nos permitiría fijar un valor para la dimensión, por ejemplo región=’Este’, el efecto es como si hubiésemos hecho “un corte” (SLICE) (región=’Este’) en el cubo de datos que representa la consulta.

La operación de DICE, consiste en focalizar la consulta en un subcubo del cubo de datos, restringiendo los valores en varias de las dimensiones.

Por ejemplo, en la consulta anterior, se podría hacer DICE para restringir los datos para las regiones Este y Centro y los meses Enero, Febrero y Marzo.

Resumiendo, una herramienta de OLAP, es una herramienta para el análisis de datos en un sistema de AD, que tiene las siguientes características:

- Presentación multidimensional de los datos objeto de análisis. En un esquema multidimensional, se presenta la actividad objeto de análisis (**hechos**) descrita por un conjunto de indicadores (**medidas**) y las **dimensiones** que caracterizan la actividad al nivel de detalle (**gránulo**) al que se almacena, estando estas dimensiones descritas por un conjunto de atributos **(atributos de dimensión**).

- Simetría en el conjunto de dimensiones: una dimensión no es más importante que otra.

- Posibilidad de definir jerarquías dentro de las dimensiones para aplicar operadores de DRILL y ROLL.

- Posibilidad de aplicar operaciones de selección (filtros) en los datos.

- Facilidades para el análisis de datos: funciones estadísticas, visualización gráfica, etc.

Debido a que las herramientas OLAP siguen una perspectiva multidimensional, las metodologías de modelado que se han desarrollado para el diseño de ADs han adoptado también esta perspectiva, y por ello se habla de "modelado multidimensional" y de "modelo multidimensional de datos".

2.2. Diseño de almacenes de datos.

2.2.1. Modelado multidimensional.

Un modelo de datos, es un conjunto de conceptos (constructores) utilizados para describir la estructura de una base de datos, los modelos de datos se clasifican en: modelos conceptuales, modelos lógicos y modelos físicos.

Los modelos conceptuales, están más próximos al usuario y son independientes de la tecnología que se vaya a utilizar, los modelos lógicos dependen del tipo de gestor de bases de datos, y los modelos físicos dependen de los sistemas comerciales particulares.

En un contexto de bases de datos, en el modelado conceptual el dominio se representa por medio de las primitivas de algún lenguaje de modelado conceptual: diagramas Entidad Relación (modelo ER), diagramas de clases (UML), etc.; una vez que se tiene el esquema conceptual éste se traduce a un modelo lógico implementado en algún Sistema Gestor de Base de Datos (SGBD) comercial (modelo relacional, modelo red o modelo jerárquico), lo que se conoce como modelado lógico y por último el esquema lógico se implementa en un SGBD comercial (modelado físico).

El diseño de un AD, sigue las mismas fases que el diseño de una base de datos operacional: conceptual, lógico y físico.

En el nivel conceptual se sigue un modelado multidimensional, en el nivel lógico un modelado dependiente de la tecnología utilizada, ROLAP (relacional) o

MOLAP (multidimensional), y en el nivel físico un modelado dependiente del gestor comercial que soporta la implementación.

El modelado multidimensional se basa en la dualidad hecho-dimensión, un **hecho** representa una actividad objeto de análisis, actividad que está caracterizada por un conjunto de **dimensiones**. En un esquema multidimensional se representa un hecho y las dimensiones que lo caracterizan. Esta representación es normalmente en forma de estrella: el hecho se representa en la parte central y las dimensiones en las puntas de la estrella. En la figura 2.9, se muestra el esquema multidimensional para la actividad de ventas en una compañía, la actividad registrada son "las ventas diarias de productos en las tiendas de la organización", en este esquema multidimensional las Ventas son los hechos y las dimensiones son: Tiempo (día), Producto (producto) y Localización (tienda).

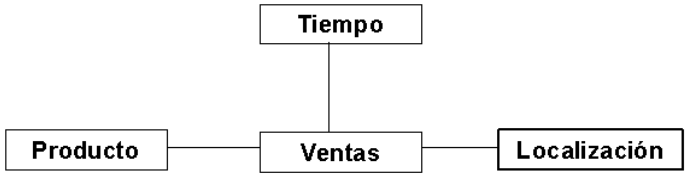


Figura 2.9. Esquema multidimensional en forma de estrella

Los hechos en un esquema multidimensional están descritos por un conjunto de atributos (medidas), usualmente de tipo numérico que describen la actividad, por ejemplo las medidas para Ventas podrían ser: importe total de la venta, número de clientes distintos que han comprado los productos, cantidad de unidades de un producto vendido, como se muestra en la figura 2.10.

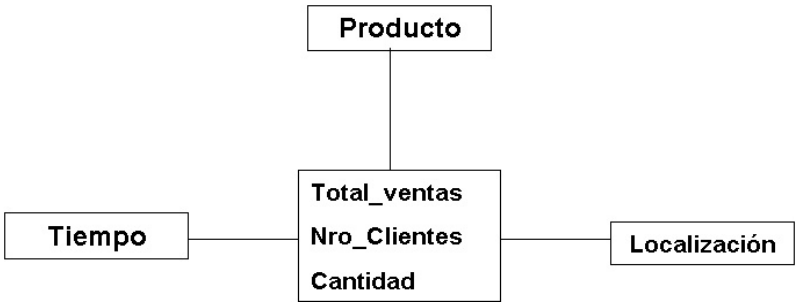


Figura 2.10. Medidas del hecho Ventas

En el esquema conceptual, las dimensiones están descritas por un conjunto de atributos que se organizan en jerarquías, en la figura 2.11 se puede observar que la dimensión Localización esta formada por cuatro atributos: tienda, ciudad, estado y región; la dimensión Tiempo esta formada por tres atributos: día, mes y año y la dimensión Producto está formada por tres atributos: producto, tipo y categoría.

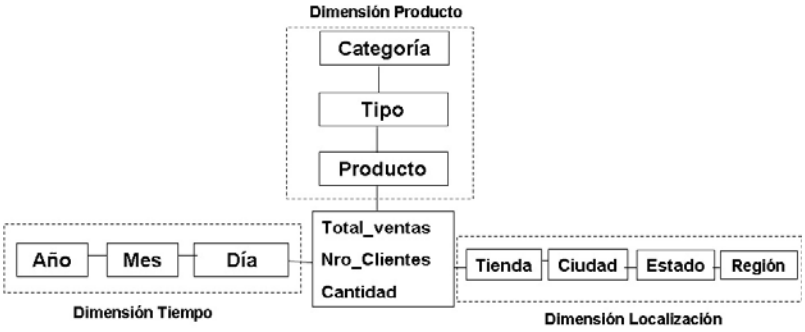


Figura 2.11. Esquema multidimensional: dimensiones con sus jerarquías

Las jerarquías entre los atributos de una dimensión representan una organización de los valores de la dimensión, que van a permitir calcular las medidas de la actividad a

distintos niveles de detalle.Por ejemplo el conjunto de datos de la figura 2.12(a), representa el total de ventas por mes para cada tienda; el nivel de detalle de este informe está determinado por el atributo *mes* de la dimensión Tiempo y por el atributo *tienda* de la dimensión Localización.

Tienda / Mes	1	2	3	4	Total
1	1000	2000	1500	3000	7500
2	2000	3000	2500	4000	11500
3	1500	4000	6000	2000	13500
4	2000	5000	7000	3000	17000
Total	6500	14000	17000	12000	49500

a)Ventas por mes y tienda

Tienda / Ciudad	Monterrey	Apodaca	Total
1	3000	4500	7500
2	5000	6500	11500
3	5500	8000	13500
4	7000	10000	17000
Total	20500	29000	49500

b)Ventas por mes y ciudad

Figura 2.12. Informes de ventas a diferentes grados de detalle

En la figura 2.12 (b) se muestra el total de ventas por mes para cada ciudad; debido a que el informe está parametrizado por el atributo *ciudad* de la dimensión Localización, se observa un menor nivel de detalle de los datos obtenidos que en el caso de la figura 2.12(a), ya que la jerarquía que existe entre los atributos *tienda* y *ciudad* indica que muchas tiendas pertenecen a una ciudad como se muestra en la figura 2.13.

TIENDA	CIUDAD
1	Monterrey
2	Monterrey
3	Apodaca
4	Apodaca

Figura 2.13. Relación jerárquica entre *tienda* y *ciudad* en la dimensión Localización

En la figura 2.14, se puede observar que la relación que existe entre los atributos *tienda* y *ciudad* es Muchos a Uno,

que es la relación normal entre los niveles de una jerarquía, sin embargo puede haber casos donde la cardinalidad de la relación entre dos niveles consecutivos sea Muchos a Muchos lo que se explicará en la sección 2.2.1.2.

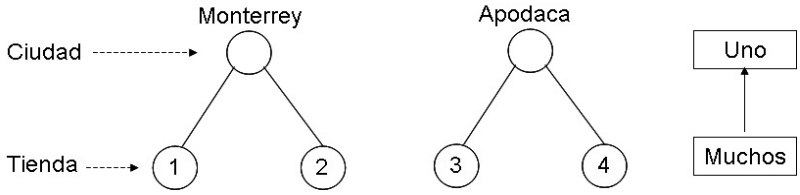


Figura 2.14. Relación 1:M entre niveles de una jerarquía

2.2.2. Medidas aditivas, semiaditivas y no aditivas.

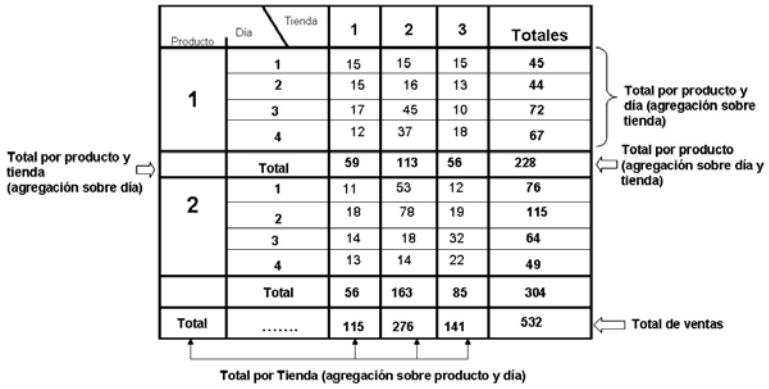
Los datos de una consulta se obtienen al agregar las medidas sobre niveles de las jerarquías, generalmente la función de agregación utilizada es la función SUM.

En algunos casos sin embargo, el valor obtenido por la función de agregación es semánticamente incorrecto, para explicar este caso utilizaremos el ejemplo anterior (figura 2.11).

Las medidas son *cantidad* que representa el número de unidades de un producto vendidas en una tienda en un día determinado, *número de clientes* que representa el número de clientes distintos que han comprado un producto en una tienda un día determinado. En el conjunto de datos de la figura 2.15, el valor de la medida *cantidad* representa la cantidad de artículos vendidos de un producto en un día en una tienda, de esta forma durante el día 1 la tienda 1 vendió 15 unidades del producto 1.

El valor que se muestra en las celdas de los totales se obtuvo al realizar la función de agregación SUM sobre las dimensiones (Tiempo, Localización y Producto), se puede observar que el valor que se tiene en estas celdas es semánticamente correcto, por ejemplo el total para el día 1, producto 1 igual a 45 es semánticamente correcto, al igual que para los demás casos por lo que la medida *cantidad* es aditiva respecto a las tres dimensiones del esquema.

Una medida aditiva se define como una medida que al aplicarle la función de agregación SUM sobre las dimensiones siempre se obtienen valores semánticamente correctos.



Producto	Día	1	2	3	Totales
1	1	15	15	15	45
	2	15	16	13	44
	3	17	45	10	72
	4	12	37	18	67
	Total	59	113	56	228
2	1	11	53	12	76
	2	18	78	19	115
	3	14	18	32	64
	4	13	14	22	49
	Total	56	163	85	304
Total	.....	115	276	141	532

Figura 2.15. Totales de la medida *cantidad*

Sin embargo, la agregación de ciertas medidas puede ser semánticamente incorrecta sobre algunas dimensiones, estas medidas son llamadas semiaditivas. Por ejemplo, el *número de clientes* es una medida semiaditiva ya que al realizar la función de agregación SUM sobre la dimensión Producto devuelve valores incorrectos, este error se puede

presentar por razones semánticas es decir por el significado de la medida o bien por el tipo de jerarquía sobre la cual se realiza la función de agregación (se explicará en la sección 2.2.1.2).

La figura 2.16, expresa el número de clientes que compraron cada día en cada tienda, por ejemplo, 13 clientes compraron durante el día 1 en la tienda 1.

Tienda / Día	1	2	3
1	13	17	12
2	17	11	17
3	14	10	15
4	16	14	12

Figura 2.16. Número de clientes por *tienda* y *día*

En la figura 2.17 se muestran los artículos que adquirió cada cliente por compra (sólo se consideran los clientes de la tienda 1 y el día 1), se puede observar que de los 13 clientes que la tienda 1 tuvo en el día 1, 10 clientes adquirieron el producto 1 y 2 en la misma compra y 3 clientes adquirieron el producto 2 y no adquirieron el producto 1. (esta información no está en el AD).

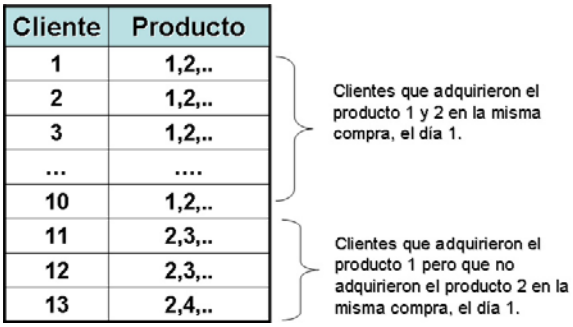


Figura 2.17. Relación de productos adquiridos por los clientes, en la tienda 1, el día 1.

En la figura 2.18, se muestra la relación de clientes que hubo por *producto*, *tienda y día* y se destaca que 10 clientes compraron durante el primer día el producto 1 en la tienda 1, y que esos mismos 10 clientes compraron el producto 2 en la misma tienda el mismo día, pero además 3 clientes distintos compraron el producto 2 en la tienda 1 el mismo día, así el total de clientes que la tienda 1 tuvo durante el día 1 fue realmente 13, (el mismo razonamiento debe hacerse para los demás casos).

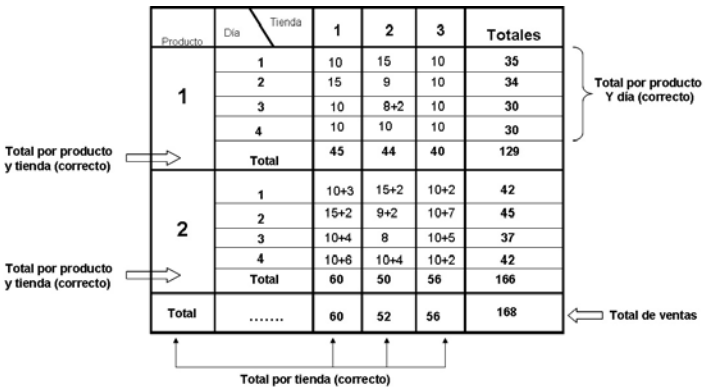
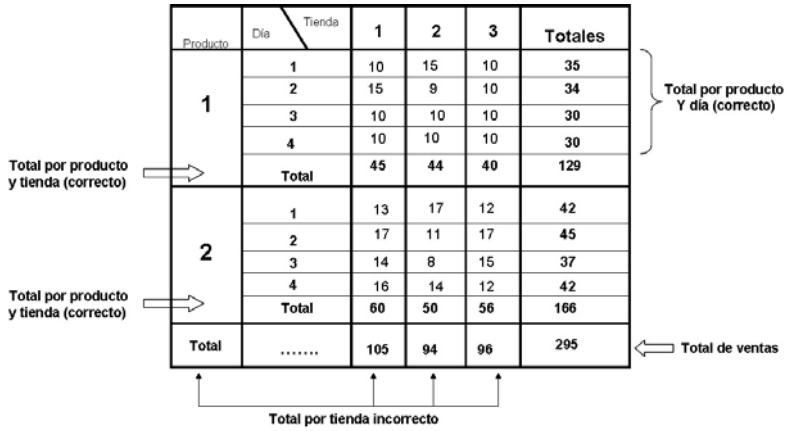


Figura 2.18. Totales esperados para la medida *número de clientes*

En base a este razonamiento el “valor esperado” al realizar la agregación es el que se muestra en el renglón total inferior de la figura 2.18, que representa el total de clientes por cada tienda, de esta forma la tienda 1 tuvo 60 clientes para el producto 1 y 2 durante los cuatro días, (la misma interpretación debe hacerse para las otras tiendas). Sin embargo el “valor que se obtendría” al realizar la agregación sobre producto y día es el que se muestra en la figura 2.19, donde se pueden observar resultados incorrectos.



Producto	Día \ Tienda	1	2	3	Totales
1	1	10	15	10	35
	2	15	9	10	34
	3	10	10	10	30
	4	10	10	10	30
	Total	45	44	40	129
2	1	13	17	12	42
	2	17	11	17	45
	3	14	8	15	37
	4	16	14	12	42
	Total	60	50	56	166
Total	.....	105	94	96	295

Figura 2.19. Informe con totales incorrectos

Los resultados incorrectos de este informe, se deben a que la herramienta OLAP desconoce que un mismo cliente ha adquirido más de un producto en la misma compra y al realizar la función de agregación SUM sobre la dimensión Producto, lo considera dos veces. Sin embargo, si realizáramos el total por producto y día (agregando sobre Tienda), el resultado mostrado sería correcto, por lo que se dice que la medida es semiaditiva: es aditiva sobre unas dimensiones y no lo es sobre otras.

Para explicar cuando una medida no es aditiva utilizaremos el siguiente conjunto de datos (figura 2.20), que representa el total de estudiantes registrados en una universidad, clasificados por departamento y año.

Depto. \ Año	2001	2002	2003	Totales
Informática	15	17	13	45
Física	10	15	11	36
Total	25	32	24	81

Figura 2.20. Relación de estudiantes registrados en la Universidad

Supongamos que queremos obtener el total de estudiantes que atendió cada departamento en los últimos tres años.
La agregación de los datos (15+17+13=45) es incorrecta suponiendo que los estudiantes pueden ser atendidos por un mismo departamento durante varios años sucesivos. Así en el año 2001, el departamento de Informática atendió a 15 estudiantes que permanecieron en el mismo departamento hasta el año 2002, por lo que en ese año 2002 se registraron sólo dos nuevos estudiantes en Informática, éstos dos estudiantes continuaron hasta el 2003, de esta forma en el año 2003 se registraron únicamente 11 estudiantes nuevos en el departamento, así el número de estudiantes total atendidos por el departamento de Informática en los tres años es 15+2+11=28.

En la figura 2.21 se puede observar que el número de estudiantes es una medida que puede agregarse sobre la dimensión Departamento (si suponemos que un mismo

estudiante no puede estar registrado en más de un departamento el mismo año) pero que no puede agregarse sobre la dimensión Año.

Depto. \ Año	2001	2002	2003	Totales
Informática	15	15+2	2+11	28
Física	10	10+5	5+6	21
Total	25	32	24	49

Figura 2.21. Medida semiaditiva sobre la dimensión Tiempo

Ahora analizaremos el conjunto de datos de la figura 2.22 suponiendo que los estudiantes pueden seleccionar materias de distintos departamentos un mismo año. Por ejemplo, un estudiante puede ser atendido por el departamento de Informática y por el departamento de Física y continuar dos años en el programa por lo que el alumno se sumaría dos veces sobre la dimensión Departamento y dos veces sobre la dimensión Año. Esta medida, "número de estudiantes registrados", es por lo tanto una mediada no aditiva.

Depto. \ Año	2001	2002	2003	Totales
Informática	14+1	17	13	28
Física	9 +1	15	11	21
Total	25	32	24	49

Figura 2.22. Resultados incorrectos por la medida no aditiva

La clasificación de las medidas en aditivas, no aditivas y semiaditivas, además de ser un aspecto semántico asociado al dominio del problema puede depender también de las propiedades de la jerarquía sobre la cual se realiza la agregación lo que se explica a continuación.

2.2.3. Clasificación de las jerarquías.

La definición de jerarquías es fundamental en el modelado multidimensional, ya que son usadas para realizar el proceso de agregación y disgregación de las medidas en los informes, por medio de las operaciones roll-up y drill-down. En la literatura, se definen tres condiciones importantes que deben cumplir las jerarquías, la condición de disyunción, la condición de completitud y la condición de cobertura.

Condición de disyunción[2]. Para explicar la condición de disyunción, utilizaremos el ejemplo de la figura 2.23, donde se muestra un esquema multidimensional para un hospital. Los hechos, representan la facturación diaria realizada a un paciente por un diagnóstico, las dimensiones son: Tiempo, Diagnóstico y Paciente.

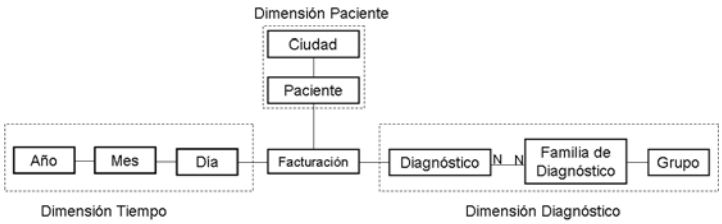


Figura 2.23. Esquema multidimensional para un hospital

[2] Las jerarquías que cumplen esta condición se llaman jerarquías estrictas.

El conjunto de datos de la figura 2.24, muestra el total facturado por diagnóstico y año, en él se observa que para el diagnóstico A se obtuvo un total de 4500 Euros durante tres años y que en el año 2001 se obtuvo un total de 6000 Euros para los tres tipos de diagnóstico.

Diagnóstico \ Año	2001	2002	2003	Total
A	1000	2000	1500	4500
B	2000	1500	2000	5500
C	3000	3000	2000	8000
Total	6000	6500	5500	18000

Figura 2.24. Informe de facturación por *diagnóstico* y *año*

Ahora consideraremos el análisis de la facturación por familia de diagnóstico y año, considerando que un diagnóstico puede pertenecer a más de una familia de diagnósticos como se muestra la figura 2.25. El informe de facturación por familias de diagnóstico y año, se muestra en la figura 2.26, donde se puede observar que es incorrecto

Diagnóstico	Familia
A	1
B	1
B	2
C	2
A	3
C	3

Figura 2.25. Clasificación de los diagnósticos por *familia*

Familia \ Año	2001	2002	2003	Total
1	1000+2000	2000+1500	1500+2000	10,000
2	2000+3000	1500+3000	2000+2000	13,500
3	1000+3000	2000+3000	1500+2000	12,500
Total	12,000	13,000	11,000	36,000

Figura 2.26. Informe de facturación por *familia* y *año*

Esto se debe a que los diagnósticos están distribuidos en subconjuntos no disjuntos (esta condición es a nivel de instancia). En la figura 2.27 se observa que un diagnóstico pertenece a más de una familia de diagnóstico, es decir existe una relación M:M entre los niveles de la jerarquía, por esto al realizar un roll-up de *diagnóstico* a *familia de diagnóstico*, se obtiene un resultado semánticamente incorrecto.

Debido a la posibilidad de obtener estos resultados incorrectos durante el análisis, el modelado multidimensional debe permitir expresar cuándo la relación de disyunción entre los atributos de una jerarquía no se cumple, indicando con esto al usuario la posibilidad de obtener resultados incorrectos. En resumen, la condición de disyunción exige que la relación entre dos niveles relacionados directamente en una jerarquía sea *1:M*.

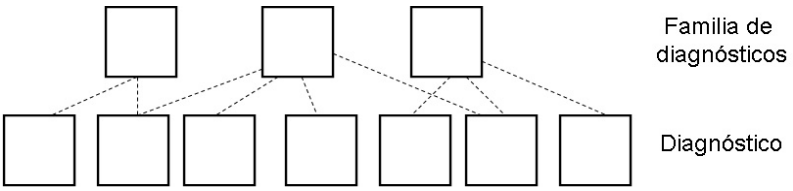


Figura 2.27. Relación jerárquica no disjunta

Condición de completitud[3]**.** Para explicar el concepto de completitud utilizaremos el esquema multidimensional de la figura 2.28, donde la medida Número de accidentes puede consultarse a través de las dimensiones Tiempo y Localización.

Figura 2.28. Esquema multidimensional sobre Accidentes.

El informe de datos de la figura 2.29, representa el número de accidentes por año que ocurrieron en distintas ciudades. Así durante el año 2001 en la ciudad Culiacán ocurrieron 150 accidentes, el valor que se muestra en los totales se obtuvo al realizar la función de agregación SUM sobre ciudad (total horizontal) y Año (total vertical).

Estado	Ciudad \ Año	2001	2002	2003	Totales
Sinaloa	Culiacán	150	300	200	650
	Mazatlán	100	200	100	400
	Los mochis	100	100	200	400
	Navolato	200	200	100	500
	Total	550	800	600	1950
Morelos	Cuernavaca	100	200	300	600
	Zacatepec	100	300	100	500
	Cuautla	200	100	100	400
	Jojutla	100	100	100	300
	Total	500	700	600	1800
Total	.....	1050	1500	1200	3750

Total por estado y año.

Total por estado y año.

Total

Total por año

Figura 2.29. Número de accidentes por *año*, *ciudad* y *estado*

[3] Las jerarquías que cumplen esta condición se llaman jerarquías completas.

Supongamos que queremos obtener el total de accidentes por estado (realizar un cambio de nivel en la dimensión Localización) y año, pero conocemos que los estados incluyen otras áreas geográficas además de las ciudades tales como zonas rurales, villas, etc., (figura 2.30), y que de estas áreas no se ha registrado información en el AD.

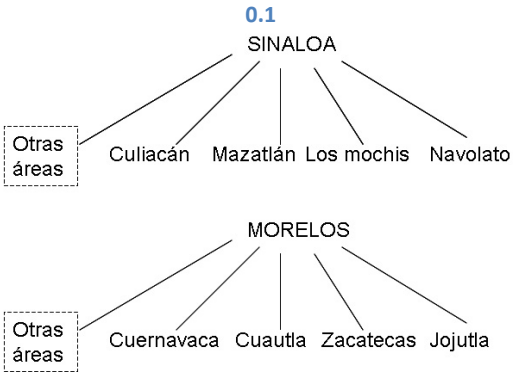


Figura 2.30. Jerarquía entre *ciudad* y *estado*

Debido a que en el informe de la figura 2.29, sólo se han considerado las ciudades el resultado que se muestra en el Total por estado y año es incorrecto. La razón por la cual es incorrecto se debe a que la relación entre los atributos estado y ciudad de la dimensión Localización no es completa, es decir los totales por estado son relativos sólo a accidentes que se han producido en ciudades que no son todos los accidentes que han tenido lugar en el estado.Esta situación se puede presentar porque las empresas ocultan información por cuestiones de privacidad o porque en la base de datos operacional no se ha almacenado dicha información [LS97].

La condición de completitud hace referencia a dos propiedades importantes de las jerarquías, la primera se refiere a que todos los elementos de un nivel no terminal existentes en el mundo real deben existir en el AD. Por ejemplo en el caso de las ciudades y otras áreas que pertenecen a un estado, es necesario que todas las ciudades y todas las áreas que pertenecen a un estado estén almacenadas, es decir que no falte ningún elemento.

La segunda propiedad hace referencia a que cada elemento de un nivel no terminal de la jerarquía debe ser asignado a un elemento del nivel superior, en el caso de las ciudades y otras áreas, todas las ciudades y todas las áreas deben ser asignadas a un estado. Estas dos condiciones se muestran en la Figura 2.31. Se puede observar, que la primera propiedad no puede controlarse en el AD ya que depende del conocimiento que se tiene sobre la carga del AD, mientras que la segunda propiedad se puede expresar con una restricción de integridad.

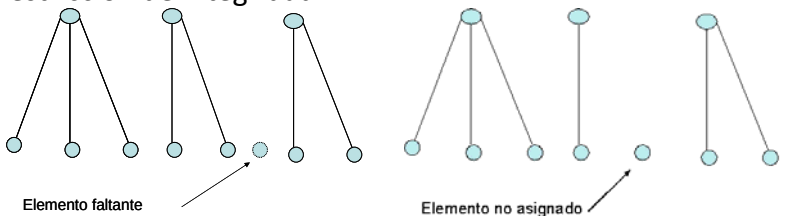


Figura 2.31. Jerarquías no completa**s**

Condición de cobertura. En el ejemplo de la figura 2.32. Se muestra la jerarquía de la dimensión Tiempo, en la que se observan dos rutas para realizar un cambio de nivel, de tal forma que es posible realizar un cambio de nivel por la

jerarquía (día→mes→año) o bien por la jerarquía (día→semestre→año).

En el esquema de la figura 2.33, se muestra las jerarquías en la dimensión Proveedor. En el esquema se destaca que existen dos rutas para realizar un cambio de nivel de *proveedor* a *país*. La primera considera los estados de cada país (proveedor→estado→país), y la segunda no los considera (proveedor→país).

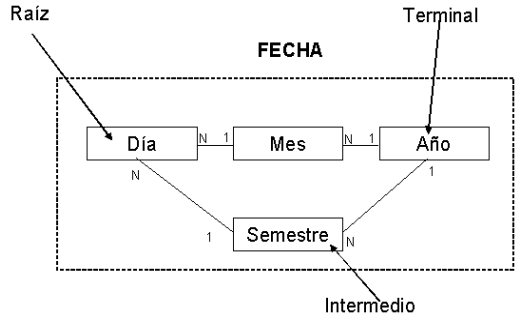


Figura 2.32. Dimensión Tiempo

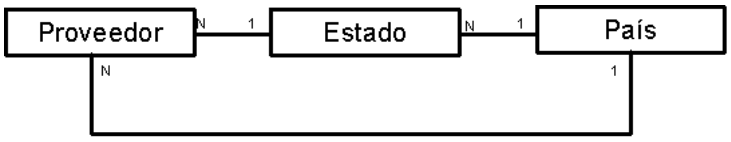


Figura 2.33. Dimensión Proveedor no cubierta

En el esquema de la figura 2.32 se observa que independientemente de la jerarquía que se elija para realizar un cambio de nivel (entre *día* y *año*), existe al menos un nivel intermedio entre ellos, por ejemplo al cambiar de *día* a *año* se encuentra el nivel intermedio *semestre*

(día→semestre→año) o mes (día→mes→año), por lo que el nivel terminal se encuentra cubierto por un nivel intermedio.

En el esquema de la figura 2.33, al hacer un cambio de *proveedor* a *país* es posible elegir la jerarquía (proveedor→país), en la cual no existe un nivel intermedio como en la jerarquía (proveedor→estado→país), diremos que la primera jerarquía no es cubierta [PDJ01].

Para mostrar los errores que se presentan al realizar un cambio de nivel cuando la dimensión no es cubierta considérese los datos de la figura 2.34.

Es importante observar que en la tabla aparecen valores nulos en el atributo *estado*, lo que significa que no se ha almacenado el estado al que pertenece el proveedor pero si se ha almacenado el país, en este ejemplo sólo se han considerado los estados de los proveedores de México.

El informe de la figura 2.35, muestra el total de compras realizadas a cada proveedor cada año. En él, se puede observar que se han considerado todos los Proveedores.

Proveedor	Estado	País
1	Sinaloa	México
2	Sinaloa	México
3	Monterrey	México
4	Monterrey	México
6	Null	España
7	Null	España

Figura 2.34. Valores en la dimensión Proveedor

Año / Proveedor	2001	2002	Total
1	1,000.00	2,000.00	3,000.00
2	2,000.00	500.00	2,500.00
3	1,000.00	1,000.00	2,000.00
4	2,000.00	1,000.00	3,000.00
5	2,000.00	1,500.00	3,500.00
6	2,000.00	3,000.00	5,000.00
7	1,000.00	2,000.00	3,000.00
Total	11,000.00	11,000.00	22,000.00

Figura 2.35. Total de compras realizadas a cada proveedor por año

En el informe de la figura 2.36(a), se observa que al realizar un cambio de nivel (roll-up) de *proveedor* a *estado*, los proveedores de España no aparecen. Mientras que al realizar un cambio de nivel roll-up) de *proveedor* a *país* (todos los proveedores son considerados, figura 2.36(b). Es decir en el informe de la figura 2.36(a) se ha perdido información relativa a algunos proveedores al no estar asignados a estados.

Así, cuando la dimensión no presenta la condición de cobertura se obtienen resultados incorrectos o confusos al realizar un cambio de nivel (roll-up).

Año / Estado	2001	2002	Total
Sinaloa	**3,000.00**	**2,500.00**	5,500.00
Morelos	**3,000.00**	**2,000.00**	5,000.00
Total	6,000.00	4,500.00	10,500.00

a) Cambio de nivel a *estado*

Año / País	2001	2002	Total
México	**6,000.00**	**6,000.00**	12,000.00
España	**5,000.00**	**5,000.00**	10,000.00
Total	11,000.00	11,000.00	22,000.00

b) Cambio de nivel a *país*

Figura 2.36. Informes con diferentes niveles de detalle

2.2.4. Relaciones Muchos a Muchos entre hechos y dimensiones.

En ocasiones la relación entre el hecho y alguna dimensión en el esquema, no es la típica relación 1:M, en varias publicaciones [KR02], [SRME01] se analizan situaciones donde la relación es M:M. Para explicar los problemas que se presentan con las relaciones M:M utilizaremos el esquema multidimensional de la figura 2.37, que representa los cobros (facturación) realizados a los pacientes de un hospital; en el esquema se muestra la relación M:M que existe entre la tabla de hechos y la dimensión Diagnóstico.

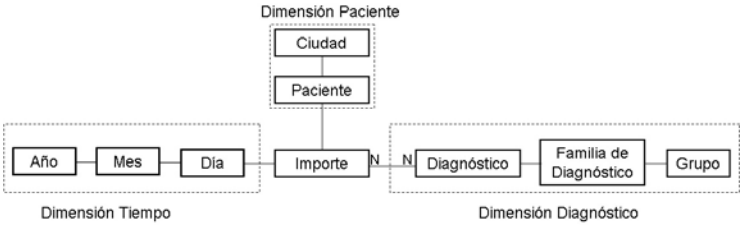


Figura 2.37. Relación M:M entre el hecho y una dimensión

En el conjunto de datos de la figura 2.38, se presentan los pagos realizados por los pacientes del hospital, y en la figura 2.39, se muestra un informe con el importe anual pagado por cada paciente. En el informe, se puede observar que el importe es una medida que puede agregarse sobre la dimensión Tiempo (año) y la dimensión Paciente. La agregación de los datos es correcta, el paciente P1, realizó dos pagos (900+400) durante el año 2000.

Paciente	Diagnóstico	Importe	Día	Año
P1	D1, D2	900	1	2000
P1	D3	400	2	2000
P2	D2, D3	900	1	2000
P2	D1	400	2	2000
P3	D1, D3	800	1	2000
P3	D2	500	2	2000
P1	D2, D3	900	1	2001
P1	D1	400	2	2001
P2	D1, D3	800	1	2001
P2	D2	500	2	2001
P3	D2, D3	900	1	2001
P3	D1	400	2	2001

Figura 2.38. Relación de pagos realizados por los pacientes

Pac. \ Año	2000	2001	Total
P1	900+400	900+400	2600
P2	900+400	800+500	2600
P3	800+500	900+400	2600
Total	3900	3900	7800

Figura 2.39. Total de ingresos anuales por *paciente*

El informe de la figura 2.40, representa los pagos anuales que realizó cada paciente por diagnóstico.

En él, se observa que el total por diagnóstico es incorrecto (total horizontal), debido a que las cantidades que se acumulan son el resultado de los diagnósticos que un paciente tuvo durante el año.

Por ejemplo, los 900 Euros de D1 en el año 2000, no son en realidad el precio del diagnóstico D1, sino el importe por los diagnósticos D1 y D2.

En la figura 2.37, también se destaca que el total por año es incorrecto debido a que el importe por diagnóstico se acumula varias veces, en este ejemplo el precio de los diagnósticos D1 y D2 se acumula dos veces durante el año 2000.

Estos resultados semánticamente incorrectos, se originan por la relación Muchos a Muchos que existe entre los hechos y la dimensión Diagnóstico.

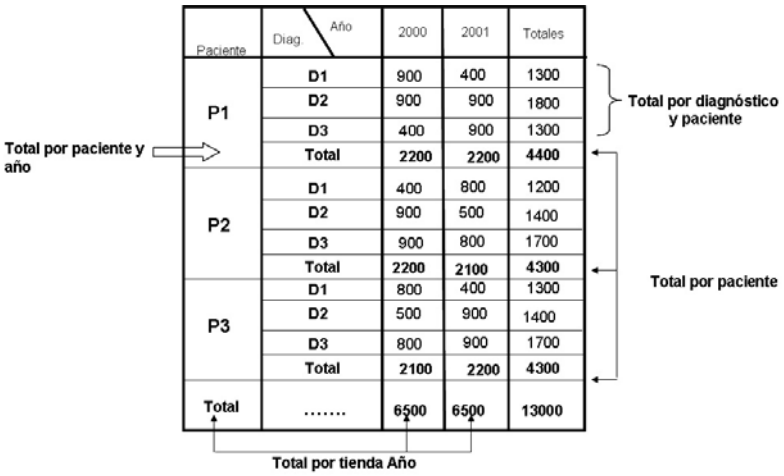


Figura 2.40. Informe de ingresos incorrecto

Capítulo 3

Introducción a MDA

En este capítulo, se presenta una introducción a Model Driven Architecture (MDA), así como a la tecnología desarrollada entorno a esta propuesta.

Posteriormente, se realiza una breve introducción a Meta Object Facility (MOF) y por último se describe el lenguaje Query View Transformation (QVT).

3.1. Introducción.

Los cambios experimentados por la tecnológica informática, hacen que los sistemas de información sean cada vez más complejos y evolutivos.

La rápida evolución tecnológica obliga a los usuarios a cambiar con frecuencia, sus sistemas de plataforma, con el alto riesgo de errores que esto supone. Esta situación parece acrecentar todavía más la eterna "crisis del software".

Para hacer frente a esta situación, la organización Object Management Group (OMG) ha lanzado una propuesta y entorno de trabajo conocida como Model Driven Architecture (MDA).

MDA, propone un nuevo paradigma de programación en el que los modelos son el elemento básico en el proceso de desarrollo de sistemas. Así, la actividad de modelado adquiere hoy en día gran importancia como técnica de abstracción para conseguir una mayor calidad del software creado y una mayor productividad en el proceso de creación.

MDA consiste en un conjunto de estándares que asisten en la creación, implementación, evolución y desarrollo de sistemas dirigido por modelos [OMG06a]. Los estándares que constituyen MDA son: Lenguaje Unificado de Modelado (UML), Meta-Object-Facility (MOF), Meta-Data Interchange (XMI) y Common Warehouse Metamodel (CWM).

Los principales objetivos de MDA son mejorar la productividad, la portabilidad, la interoperabilidad y la reutilización de sistemas. Para ello el desarrollo del sistema

se basa en los modelos de datos y en la transformación entre ellos, transformación dirigida por medio de un conjunto de reglas.

Una de las ideas básicas de MDA, es que todo modelo de datos tiene asociado un metamodelo. Cada metamodelo, define un lenguaje específico y todos los metamodelos se basan en un metamodelo único. En MDA, este metamodelo es MOF (figura 3.1). MOF, es el metamodelo usado para definir metamodelos.

La especificación de las reglas de transformación, se realiza por medio de un lenguaje de transformación que permite definir patrones de transformación entre los metamodelos y en el que se establecen las correspondencias entre el metamodelo fuente y el metamodelo destino. Un patrón de transformación, captura las propiedades comunes para una familia de transformaciones a nivel de modelo. Las reglas de transformación, establecen las correspondencias entre un metamodelo fuente y un metamodelo destino y permiten la transformación de modelos. El motor de transformación, aplica las reglas de transformación al modelo fuente y devuelve el modelo destino.

3.2. Meta Object Facility (MOF).

Un modelo, es una descripción parcial o total de un sistema escrito en un lenguaje de modelado. El hecho de que un modelo este escrito en un lenguaje (bien formado) es importante en el contexto de MDA, ya que este hecho garantiza que el modelo esta asociado a una sintaxis y una semántica bien definidas.

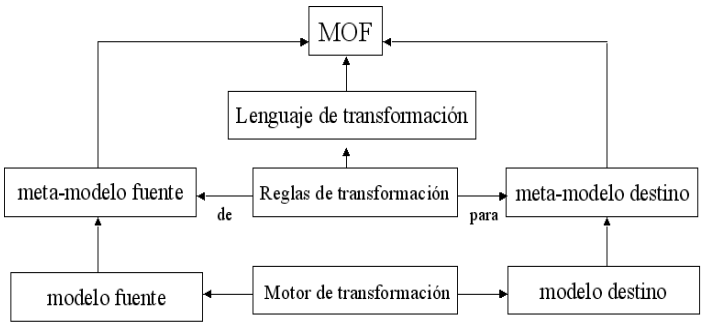


Figura 3.1. Elementos que componen MDA

En MDA, un lenguaje de modelado esta asociado a un metamodelo específico, esta característica permite la interpretación automática de cualquier modelo, lo que es fundamental para la transformación de modelos.

Un metamodelo, es un mecanismo que permite la definición formal de modelos. Para la definición de metamodelos, OMG propone el uso de MOF. Para la definición formal de la sintaxis abstracta del conjunto de constructores de un lenguaje de modelado, se puede utilizar MOF; es decir MOF es un lenguaje que puede ser utilizado para describir lenguajes de modelado.

Estos lenguajes de modelado pueden ser considerados instancias de MOF, puesto que lo que proporciona MOF son los constructores y mecanismos mínimos necesarios para definir metamodelos (lenguajes de modelado), es decir, los elementos que van a constituir un lenguaje dado y las relaciones entre dichos elementos.

MOF utiliza cinco constructores básicos para definir lenguajes de modelado:

- **Clases**: usadas para definir tipos de elementos del metamodelo.
- **Generalización**: usada para definir relaciones de herencia entre las clases.
- **Atributos**: usados para definir las propiedades de las clases.
- **Asociaciones**: usadas para definir relaciones entre las clases.
- **Operaciones**: usadas para definir procedimientos en el ámbito de una clase.

El papel principal de MOF dentro de MDA, es proporcionar los conceptos y herramientas para razonar sobre lenguajes de modelado. Gracias a ello, es posible definir las transformaciones sobre los metamodelos de una manera estándar.

3.3. La arquitectura de cuatro niveles de MDA.

Como se mencionó anteriormente, la intención de MOF es proporcionar un marco de trabajo uniforme para la descripción de metamodelos. Para lograrlo, la OMG propone una jerarquía de cuatro niveles de modelado que permiten representar la información a diferentes niveles de abstracción. En la terminología propuesta por la OMG estos niveles son: M0, M1, M2 y M3.

- ***Nivel M0: instancias***. En el nivel M0, se encuentran todas las instancias del sistema de información, es decir los objetos de la aplicación. En esta capa, no se hace referencia a las clases ni atributos, sino a las entidades físicas del sistema.

- ***Nivel M1: modelo del sistema*[4].** Por encima de la capa M0 se localiza la capa M1. Esta capa, representa el modelo de una aplicación. Los conceptos del nivel M1 representan categorías de las instancias de M0, es decir: cada elemento de M0 es una instancia de un elemento de M1.
- ***Nivel M2: metamodelo.*** Análogamente a la relación entre M0 y M1, los elementos del nivel M1, son a su vez instancias del nivel M2. Esta capa recibe el nombre de capa de metamodelo. El nivel M2 lo forman todos aquellos metamodelos que son instancias de MOF.
- ***Nivel M3: meta-metamodelo.*** De igual forma, podemos ver los elementos de M2 como instancias de otra capa superior, la capa M3, la capa del meta-metamodelo. Este nivel, está formado por un lenguaje capaz de definir nuevos metamodelos, es decir MOF.

Se podrían añadir más capas a las ya descritas, lo cual no sería útil. En lugar de definir la capa M4, OMG estableció que todos los elementos de M3 se puedan auto-definir con instancias de conceptos de M3.

La figura 3.2, muestra un ejemplo completo donde se aprecia la relación existente entre las capas de modelado. En la capa M0, se muestran las entidades que almacena los datos (*nombre ="Manolo")*. En el nivel M1, aparece la entidad *Empleado* con el atributo *nombre* que representa a

[4] *El esquema conceptual de la base de datos se ubica en el nivel M1(esquema modelo)*

todos los empleados de la aplicación con su propiedad "nombre".

La entidad *Empleado* del nivel M1, es una instancia de la metaclase *Class* del metamodelo de UML en el nievel M2 y el atributo *nombre* de *Empleado*, es una instancia de la metaclase *Attribute* del metamodelo UML.

Finalmente en la figura 3.2, se muestra que las clases del nivel M2 son instancia de la clase MOF *Class* del nivel M3.

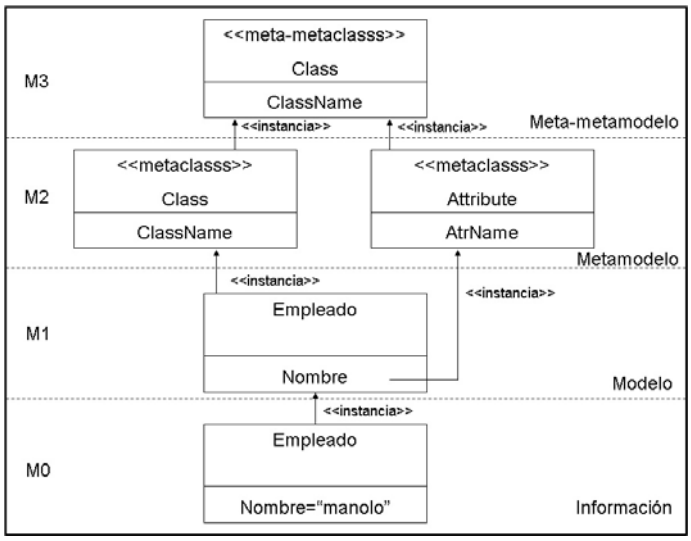


Figura 3.2. Niveles de abstracción de MDA

3.4. Niveles de abstracción en MDA.

La historia del desarrollo de software puede verse como una incorporación sucesiva de niveles de abstracción, en un recorrido que acerca la programación a la representación del

espacio del problema y la aleja del ámbito del espacio de la solución [MP04].

El objetivo de la iniciativa MDA, es la separación de la lógica de las aplicaciones de la plataforma software en la que dicha lógica vaya a ser implementada.

Esta separación, permite reducir el impacto que la evolución de las tecnologías tiene en el desarrollo de aplicaciones, al permitir que una misma especificación pueda ser implementada en diferentes plataformas software. Por otro lado, traslada el conocimiento de una aplicación desde el código fuente (enfoque tradicional) a la especificación.

El modelo se convierte en el elemento más valioso puesto que a partir de él, mediante una serie de transformaciones se puede obtener el código de la aplicación.

El desarrollo de aplicaciones en MDA se basa en la idea de capturar los requisitos de usuario, en un modelo independiente del computador llamado *Computation Independent Model* (CIM) para refinarlo posteriormente a un modelo llamado *Platform Independent Model* (PIM).

El PIM tiene por objetivo modelar las aplicaciones que se van a desarrollar en un modelo independiente de los detalles de implementación o de la plataforma software en la que la aplicación vaya a ejecutarse, finalmente estos PIMs, son transformados en otros modelos que incorporan detalles de implementación llamados *Platform Specific Model* (PSM). En la figura 3.3, se muestra la relación entre modelos.

La ventaja más importante de este enfoque, es el poder definir transformaciones automáticas entre un PIM y distintos PSMs o entre distintos PIMs de forma que a partir del PIM, del PSM y de un conjunto de reglas de transformación, se pueda obtener la implementación del sistema de la forma más automatizada posible.

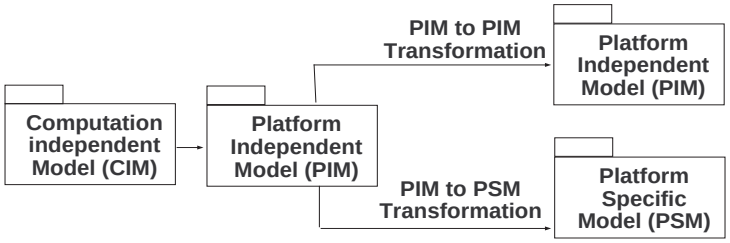


Figura 3.3. Clasificación de modelos en MDA

A continuación se describe cada modelo:

- ***CIM*.-** Representa una vista del sistema. Un CIM oculta detalles sobre la estructura del sistema y en ocasiones es llamado también el modelo del dominio del problema.

  El CIM, representa los requisitos de usuario y en su definición se emplea un vocabulario que es familiar para todos los involucrados en el desarrollo del sistema.

  El CIM, juega un papel importante durante el desarrollo del sistema ya que reduce la brecha entre aquellos que son expertos del dominio del problema y aquellos que son expertos en el diseño y construcción del sistema.

- ***PIM*.-** Representa la estructura, funcionalidad y restricciones del sistema independientemente de la plataforma tecnológica sobre la que se va a implementar.

  Al no incluir detalles específicos de una tecnología de implementación, facilita la comprensión del sistema por parte de los usuarios y la implementación del sistema en distintas plataformas.

- ***PSM*.-** Es la representación de un sistema, con detalles específicos de la plataforma en la que será implementado. Se genera a partir del PIM, así que representa el mismo sistema pero a distinto nivel de abstracción.

3.5. Query View Transformatios (QVT).

Hasta ahora, todo lo expuesto sobre MDA ha sido en torno al trabajo realizado en la definición de modelos en distintos niveles de abstracción, siendo éstos el fundamento de todo el desarrollo automático basado en modelos. Sin embargo uno de los principales retos de este estilo de programación, es la transformación entre modelos. Por ello la OMG emitió una solicitud de propuestas para suplir la falta de un estándar de transformación en el ámbito de MOF, el QVT (Query/Views/Transformations) [OMG04]. Las propuestas tenían que ser compatibles con los estándares de OMG: UML, MOF, CWM y se pedía un lenguaje que tuviera las siguientes características:

- Permitiera definir transformaciones entre metamodelos definidos bajo MOF.

- Permitiera crear la vista de un modelo.
- Fuera declarativo.

En resumen, QVT es un estándar que define la manera en la cual se llevan a cabo las transformaciones entre modelos cuyos metamodelos se basan en MOF. Este estándar consta de tres componentes [OMG05b]: consultas, vistas y transformaciones.

3.5.1. Consultas.

Una consulta, es una expresión sobre un modelo. El resultado de la consulta, es una o mas instancias de los tipos definidos sobre el metemodelo fuente. Por ejemplo, si definimos una consulta sobre un modelo UML que regresa todas las clases del tipo persistente (*select (c|c.kind = "persistente*"), el resultado podría ser una colección de las instancias tipo clase. Lo mismo sucede para la consulta que regresa los atributos de un modelo UML, cuyo nombre sea igual a "*ejemplo*" *(select (a|a.name="ejemplo")* → *NotEmpty*, el resultado seria un valor booleano (figura 3.4).

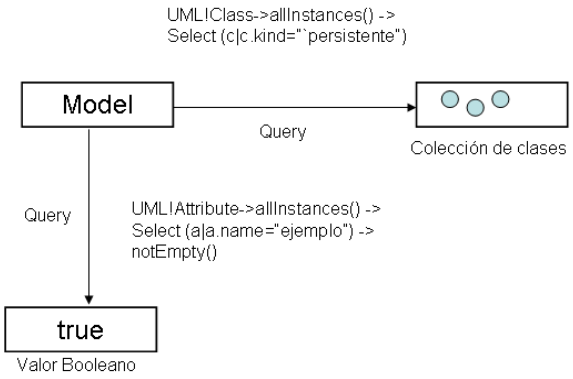


Figura 3.4. Consultas realizadas sobre un modelo

Como lenguaje de consulta, se emplea Object Constraint Language (OCL) [OMG06b]. Con esta elección se obtienen los siguientes beneficios:

a) La comunidad de usuarios ya está familiarizado con el lenguaje.

b) No es necesario realizar un esfuerzo adicional para definir un nuevo lenguaje.

c) Actualmente, OCL está implementado en muchas herramientas.

3.5.2. Vistas.

Una vista, es un modelo derivado de otro modelo (modelo base). Una vista, no puede ser modificada independientemente del modelo del cual se deriva.

Los cambios que se realicen en el modelo base, ocasionan cambios en la vista. Una vista se crea mediante un proceso de transformación (figura 3.5).

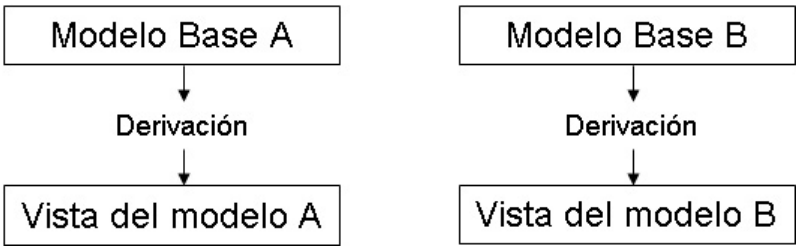


Figura 3.5. Vistas creadas a partir de modelos

3.5.3. Transformaciones.

Un proceso de transformación es la generación automática de un modelo destino a partir de un modelo fuente.

Una transformación, se compone de un conjunto de reglas de transformación que juntas describen la forma como se transforma un modelo fuente en un modelo destino.

Una regla de transformación consiste en una descripción de cómo uno o más constructores de un modelo fuente pueden ser transformados en uno o más constructores de un modelo destino.

En la figura 3.6, se muestran como un PIM es convertido a un PSM por medio de un proceso de transformación.

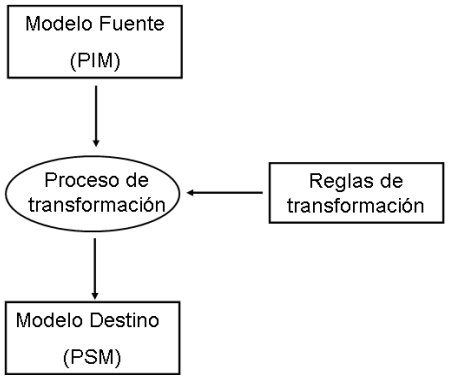


Figura 3.6. Proceso de transformación

3.5.4. Tipos de transformaciones.

En [MSUW04] las transformaciones se clasifican en dos tipos: verticales y horizontales.

Esta clasificación, depende del nivel de abstracción al que pertenecen los modelos destino:

- **Transformación horizontal**.- Se presenta cuando el modelo fuente y destino se encuentran en el mismo nivel de abstracción (figura 3.7).

  Este tipo de transformación, permite relacionar modelos de distintos dominios, pero al mismo nivel de abstracción.

Figura 3.7. Transformación horizontal

- **Transformación vertical**.- Se presenta cuando el modelo fuente y el modelo destino se encuentran a distintos niveles de abstracción.

  La transformación vertical, se realiza cuando es necesario tener una definición del modelo más cercano a la tecnología de implementación, por ejemplo de un PIM a PSM o bien de PSM a código (figura 3.8).

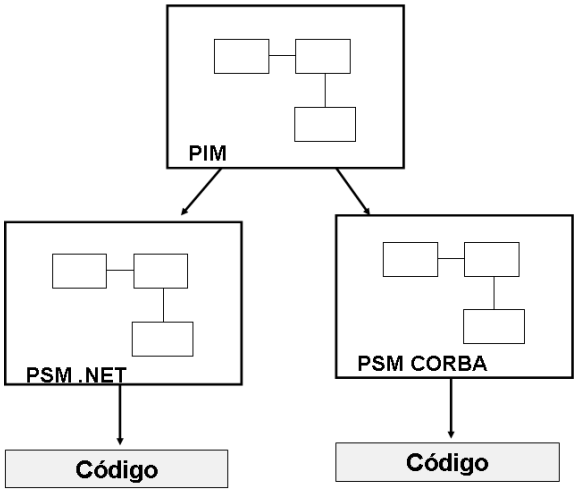


Figura 3.8. Transformación vertical

Capítulo 4

Diseño conceptual de almacenes de datos: estado del arte

En este capítulo, se presenta una revisión del estado del arte en el área del diseño conceptual de almacenes de datos.

Las metodologías que aquí se presentan, son clasificadas en tres grupos según la aproximación que se sigue para obtener el esquema conceptual del almacén de datos: metodologías dirigidas por los datos, metodologías dirigidas por los requisitos de usuario y metodologías compuestas.

4.1. Metodologías de diseño de almacenes de datos dirigidas por los datos.

En esta sección se presenta un análisis de las metodologías de diseño de AD que derivan el esquema del AD a partir del esquema de las bases de datos operacionales, apoyándonos en el ejemplo presentado en 4.1.1. De cada propuesta se analizarán las siguientes caracterísiticas:

1. Aditividad.
2. Jerarquías disjuntas.
3. Jerarquías completas.
4. Jerarquías cubiertas.
5. Relaciones Muchos a Muchos entre hechos y dimensiones.
6. Categorización de dimensiones.

4.1.1. Ejemplo: Cadena de puntos de venta.

El diagrama Entidad Relación de la figura 4.1, modela la actividad de *Ventas* en una cadena de tiendas (o supermercados). En el diagrama ER las ventas son representadas por la entidad Ticket, en la cual se almacena el importe de la venta y la fecha en que se efectuó.

La relación Línea representa los artículos de cada venta, esta relación almacena la cantidad de artículos vendidos, el precio unitario de cada artículo y el total de la línea. La información que se almacena de cada Artículo es la descripción del artículo y el precio de coste, además se almacena la información relacionada con el fabricante y la categoría de cada artículo como se muestra en el diagrama.

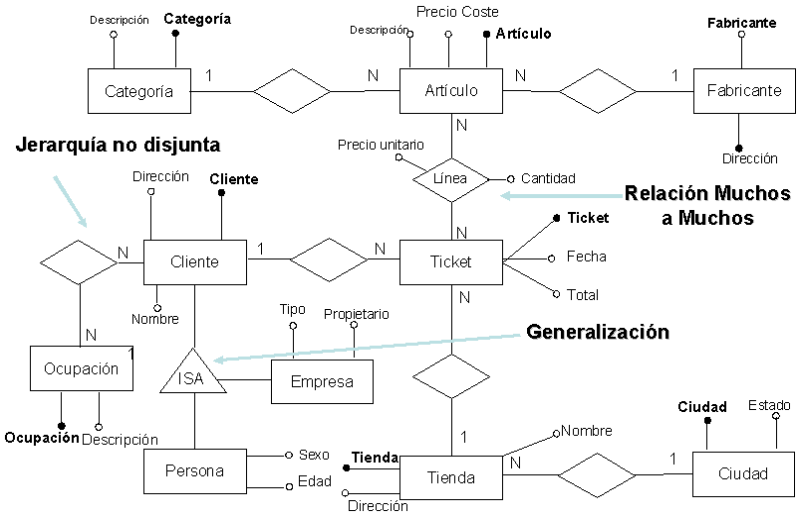


Figura 4.1. Diagrama ER

Los clientes de las tiendas (entidad Cliente) se encuentran clasificados como Empresa o Persona por lo que existe una generalización entre esas entidades. Por medio de la relación entre Ticket y Tienda, es posible identificar las ventas realizadas por cada tienda. La entidad Ocupación, almacena las distintas ocupaciones que han tenido los clientes a través del tiempo.

4.1.2. Multidimensional Fact Model (DFM).

Golfarelly, M. y Dario, M. [GMR98a] [GMR98b] proponen el modelo DFM, el cual permite hacer una representación de los hechos y dimensiones con una notación gráfica propia, además proponen una metodología semiautomática para obtener un esquema multidimensional a partir de un diagrama ER. Un esquema en DFM se define como una colección de esquemas de hechos, cuyos elementos básicos

son los hechos, los atributos, las dimensiones y las jerarquías. Otros elementos que pueden ser representados en el esquema de hechos son las propiedades de aditividad entre las medidas y las dimensiones.

La metodología que proponen para derivar un esquema multidimensional a partir de un esquema ER consiste en:

1. Definir los hechos.
2. Por cada hecho.
    a. construir el árbol de atributos.
    b. podar o insertar ramas en el árbol de atributos.
    c. definir las dimensiones.
    d. definir los atributos de hechos.
    e. definir las jerarquías.

1. Definir los hechos**.-** La metodología establece que un hecho en un diagrama ER corresponde a una entidad o a una relación n-ary entre entidades con una frecuencia alta de actualización. Cuando un hecho es una relación n-ary se debe transformar el diagrama ER de tal forma que en él sólo aparezcan relaciones Muchos a Uno o relaciones Uno a Uno como un caso particular.

   En el ejemplo de Ventas, un hecho puede ser la relación Línea que tiene cardinalidad Muchos a Muchos con las entidades Ticket y Artículo. El diagrama ER modificado se muestra en la figura 4.2.

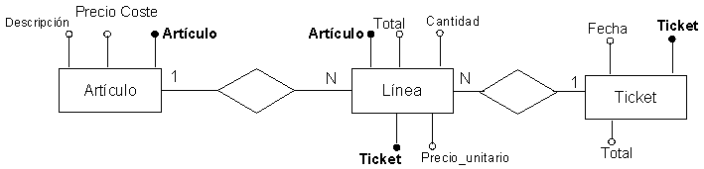


Figura 4.2. Relación modificada

a) Construir el árbol de atributos**.-** Una vez identificados los hechos se construye un árbol de atributos para cada hecho, donde la raíz del árbol esta formada por un hecho y cada nodo corresponde a un atributo de las entidades del esquema ER relacionadas de manera directa o indirecta a la entidad que representa el hecho. La construcción del árbol se realiza de manera automática al aplicar un procedimiento recursivo donde la condición para agregar un nodo al árbol consiste en considerar las entidades del diagrama conectadas directa o indirectamente con el hecho por medio de una relación “Muchos a Uno”.

En la figura 4.3, se puede observar que la raíz del árbol esta formada por el hecho Línea y que los demás nodos del árbol corresponden a atributos de las entidades asociadas al hecho por relaciones “Muchos a Uno”. La entidad Ocupación no aparece en el diagrama, debido a la cardinalidad Muchos a Muchos que existe entre entidad Cliente y la entidad Ocupación. (no cumple con la cardinalidad “Muchos a Uno”, requerida por el procedimiento), es decir el modelo no permite representar las jerárquicas que no son disjuntas.

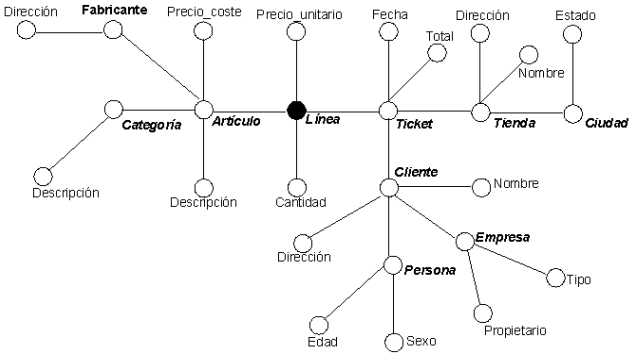


Figura 4.3. Árbol de atributos derivado del Diagrama ER

b) Podar o insertar ramas al árbol de atributos**.-** Debido a que no todos los atributos representados en el árbol de atributos son de interés para el diseño del AD, estos pueden ser “podados” o “injertados” en el árbol con la finalidad de eliminar niveles de detalle innecesarios.

Durante el proceso de “podado” los atributos son eliminados del árbol y un “injerto” se realiza cuando un nodo del árbol es “podado” pero se requiere preservar a sus descendientes, por lo que estos pasan a ser hijos del vértice padre del nodo eliminado.
En el diagrama de la figura 4.4, se puede observar que se eliminaron los atributos Precio_coste, el nodo Fabricante, y el nodo Ticket por lo que el atributo Fecha del nodo Ticket pasa a ser parte del nodo Línea (que es el nodo padre de Ticket).

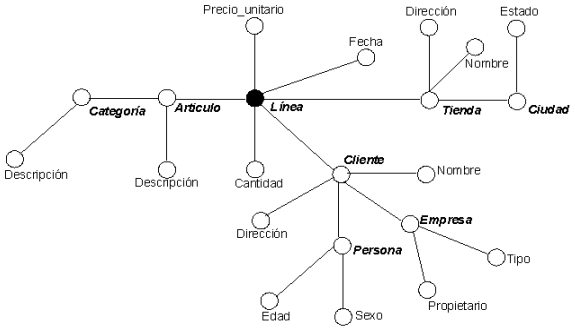


Figura 4.4. Árbol de atributos modificado

c) Definir las dimensiones**.-** Se consideran dimensiones los nodos de la raíz del árbol que correspondan a una entidad del diagrama ER o el atributo Fecha. Las dimensiones identificadas en el diagrama de la figura 4.4 son: Artículo, Clientes, Tienda y Fecha, en la figura 4.5, se pueden observar las dimensiones identificadas.

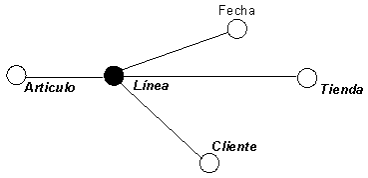


Figura 4.5. Dimensiones identificadas

d) Definir los atributos de hechos.- Los atributos de los hechos son típicamente contadores de instancias o funciones de agregación de la tabla de hechos como SUM, AVG, MAX, MIN y COUNT.

En este punto el autor recomienda realizar un glosario en el cual se asocie a cada atributo de

hechos una expresión de agregación que describa como puede calcularse a partir de los atributos del esquema ER.

Para este ejemplo, consideraremos sólo los atributos de hechos Cantidad Vendida y Número de Clientes, en la figura 4.6 se muestra la función de agregación asociada a cada uno de ellos.

Cantidad Vendida = Sum(Linea.Cantidad)
No. Clientes = Count(Linea)

Figura 4.6. Funciones de agregación asociadas a los atributos de hechos

e) Definir las jerarquías**.-** El último paso consiste en definir las jerarquías sobre las dimensiones.

En cada jerarquía, los atributos deben ser organizados en el árbol de forma que las relaciones "Muchos a Uno" se conserven.

También en este punto se puede realizar un "podado" y un "injerto" de las relaciones con cardinalidad 1:1. Por ejemplo la generalización entre Cliente, Empresa y Persona de la figura 4.1, tiene cardinalidad 1:1, por lo que se debe de eliminar la entidad Empresa y la entidad Persona del árbol ("Podado") y sus atributos "Injertarlos" al nodo padre de los nodos eliminados (Cliente), de esta forma los atributos de los nodos eliminados pasan a ser parte del nodo Cliente como se muestra en la figura 4.7.

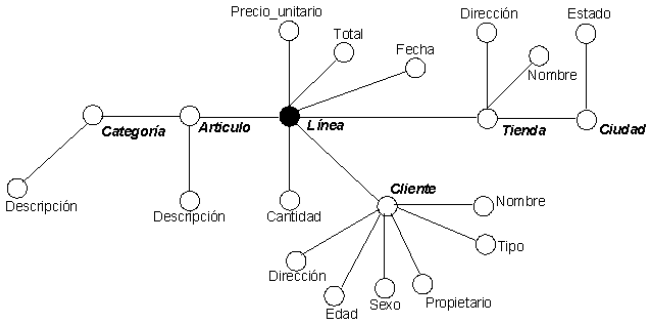


Figura 4.7. Árbol de atributos después de eliminar la generalización

Algunos atributos del árbol serán considerados atributos descriptivos de una dimensión, y no afectarán el nivel de detalle de la consulta pero permitirán aplicar filtros.

Las medidas se consideran aditivas sobre todas las dimensiones por omisión, cuando una medida no sea aditiva sobre una dimensión se indica por medio de una línea punteada. Una vez definidas las jerarquías, se realiza el diagrama MD donde los hechos son representados por medio de un cuadrado y las dimensiones alrededor de él por medio de líneas y círculos como se muestra en la figura 4.8.

Después de aplicar la metodología al ejemplo se puede observar que el modelo DFM contempla todos los elementos básicos del modelado multidimensional como son: hechos, dimensiones, jerarquías, atributos descriptivos, y medidas asociadas a los hechos, además permite representar de manera gráfica los atributos aditivos y los atributos no aditivos.

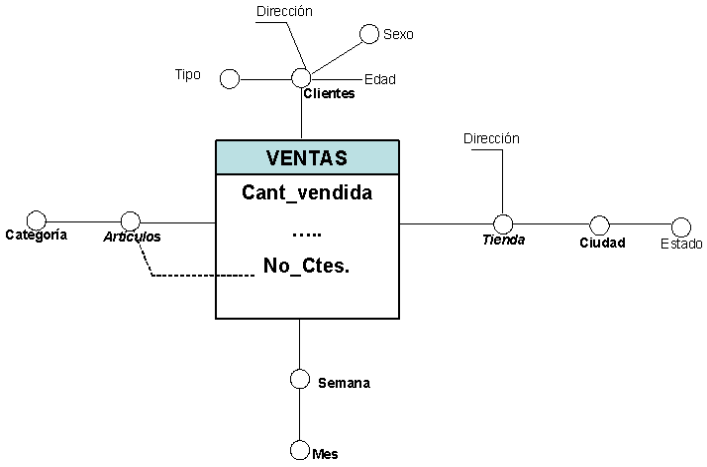


Figura 4.8. Diagrama de hechos del esquema de Ventas

Las medidas se definen al aplicar una función de agregación sobre un atributo numérico y la categorización de dimensiones es tratada como una relación 1:1. Sin embargo no es posible representar jerarquías disjuntas, jerarquías completas y la relación Muchos a Muchos entre hechos y dimensiones. En la figura 4.9 se muestra un resumen de las propiedades que se pueden modelar.

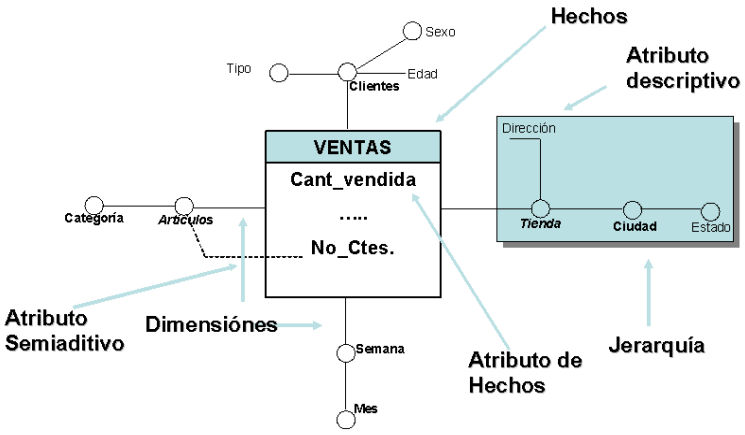


Figura 4.9. Propiedades contempladas en el modelo DFM

4.1.3. Modelo Multidimensional (MD).

Cabibbo y Torlone [CT98] proponen el método de diseño *MD*, que definen como un modelo lógico para sistemas OLAP, sin embargo los autores mencionan que es independiente de cualquier implementación, por lo que lo ubican en el nivel conceptual.

El método de diseño que proponen construye un esquema *MD* a partir de una base de datos operacional existente, el esquema *MD* consiste de un conjunto finito de Dimensiones y un conjunto finito de *F-Tables (*Hechos*),* donde las dimensiones son categorías sintácticas que permiten especificar múltiples caminos para la búsqueda de información y cada dimensión se organiza en una jerarquía de niveles correspondientes.

Un nivel puede tener descriptores asociados a él. Dentro de las dimensiones, los valores de diferentes niveles son

relacionados a través de funciones roll-up y drill-down. Las *F-Tables* son funciones de coordenadas simbólicas (definidas sobre una combinación particular de niveles) a medidas que son usadas para representar datos.

La metodología propuesta para construir una base de datos MD a partir de los diagrama ER consiste en los siguientes pasos:

1. Identificar los hechos y las dimensiones.
2. Reestructurar el diagrama Entidad Relación.
3. Derivar un grafo dimensional.
4. Trasladar el grafo al modelo MD.

1.- La primera actividad, consiste en realizar un análisis detallado del diagrama ER para identificar y seleccionar los hechos, las dimensiones y las medidas. Un hecho puede ser un atributo numérico, una entidad o relaciones con cardinalidad Muchos a Muchos. En el ejemplo (figura 4.1), se pueden considerar como hechos: la relación Línea (ya que tiene cardinalidad Muchos a Muchos), y el atributo Precio de Coste de la entidad Artículo figura 4.10.

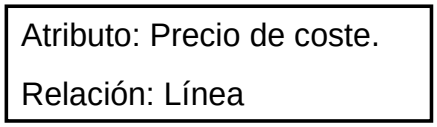


Figura 4.10. Hechos identificados en el diagrama ER

Las medidas son una propiedad atómica de un hecho que se intenta analizar, por lo general un atributo numérico o un contador de instancias. Las medidas pueden ser los Artículos Vendidos y el Número de Clientes.

Las dimensiones son un subesquema del diagrama ER que permite el análisis de un hecho, las dimensiones se identifican en el diagrama por medio de relaciones Muchos a Uno, excluyendo las relaciones Uno-Uno y Muchos a Muchos, las dimensiones identificadas en el diagrama se muestran en la figura 4.11.

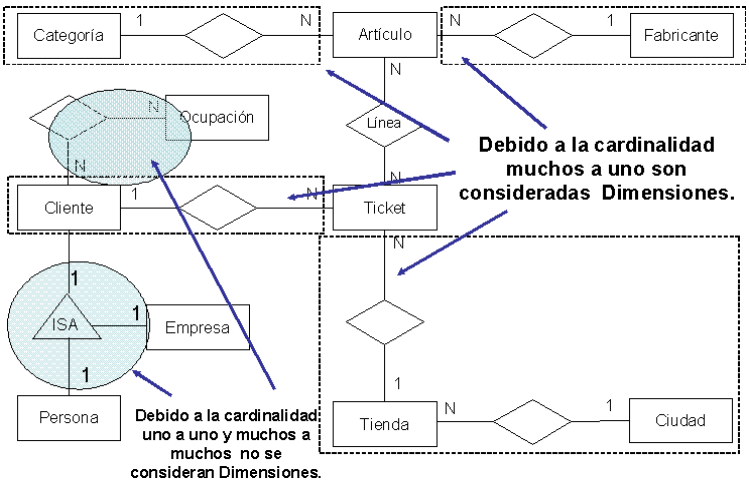


Figura 4.11. Elementos identificados en el esquema ER

2.- Una vez identificados los elementos básicos del modelo multidimensional se realiza una reestructuración del diagrama ER con la finalidad de describir los hechos y dimensiones de manera más explícita, la reestructuración se compone de tres actividades:

a) Representar los hechos como entidades**.** Cuando el hecho identificado es una relación o un atributo deben de representarse como una entidad. Por ejemplo, el atributo Precio de Coste el cual fue identificado como hecho debe transformase en una

entidad. Este atributo puede ser transformado a la entidad PRECIO DE COSTE, agregando una relación Uno a Uno entre la nueva entidad y la entidad Artículo, como se muestra en la figura 4.12.

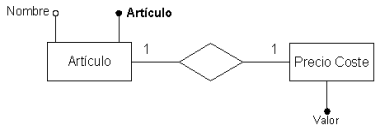


Figura 4.12. Reestructuración del atributo Precio de Coste

Las relaciones Muchos a Muchos identificadas como hechos deben representarse como una nueva entidad y dos relaciones Muchos a Uno.

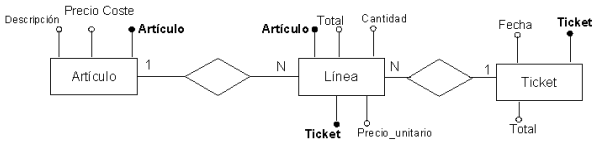


Figura 4.13. Reestructuración de la relación muchos a muchos

b) Agregar dimensiones**.** En esta etapa, las dimensiones que no aparecen de manera explícita en el diagrama pueden ser agregadas de acuerdo a la experiencia que tiene el equipo de desarrollo sobre el problema.

Por ejemplo, la dimensión Fecha puede agregarse a la entidad Precio de Coste lo cual permitiría realizar análisis de Costes, el representar que un Artículo tiene un Coste distinto en distintos periodos de tiempo cambia la cardinalidad entre Precio de Coste y Artículo, como se muestra en la figura 4.14.

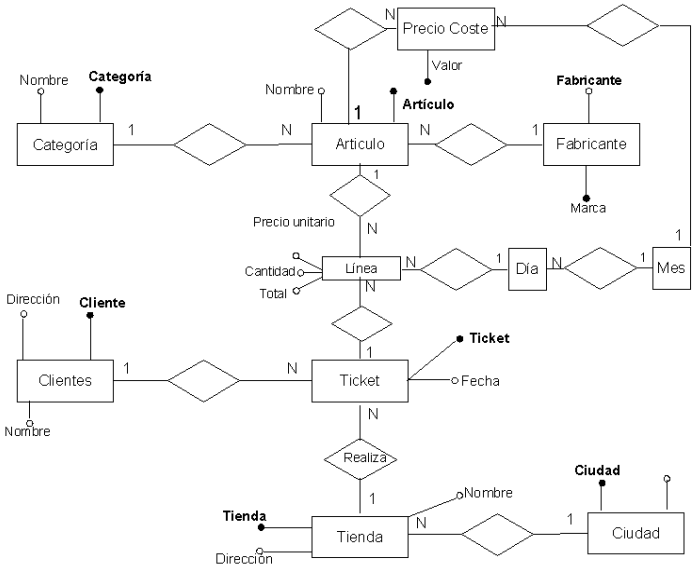


Figura 4.14. Diagrama ER modificado

Por medio de un grafo dimensional (figura 4.15) se representan los hechos y las dimensiones el diagrama ER reestructurado, en el grafo multidimensional se pueden identificar cuatro tipos de nodos: nodos de hechos, nodos de niveles, nodos descriptivos y nodos de medidas.

Este modelo cumple con las características generales de los modelos conceptuales, a pesar de que contempla conceptos básicos del modelado multidimensional, no es posible definir cuándo un atributo es aditivo o no sobre una dimensión, no permite representar la categorización de dimensiones, la disyunción en las jerarquías, las relaciones Muchos a Muchos entre hechos y dimensiones ni las jerarquías no cubiertas.

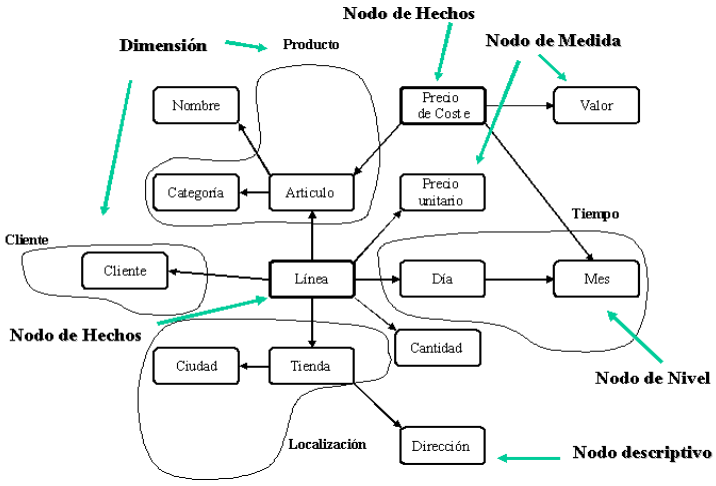


Figura 4.15. Grafo multidimensional

4.1.4. Modelo MER.

Cassandra y Karen [PD02] presentan un algoritmo y una metodología para obtener un conjunto de esquemas multidimensionales MER a partir de las bases de datos operacionales.

El algoritmo para derivar esquemas multidimensionales a partir del esquema ER del operacional sigue los siguientes pasos:

1. Encontrar las entidades con atributos numéricos y crear un nodo de hechos por cada entidad identificada.
2. Asociar a cada nodo de hechos los atributos numéricos de la entidad.

3. Crear niveles de fecha o tiempo por cada tabla de hechos.
4. Crear un nivel (en la dimensión) que contenga los atributos restantes de una entidad (no numérico, no llave y no tipo fecha).
5. Examinar de manera recursiva las relaciones entre las entidades para agregar niveles a cada dimensión.

El resultado de este algoritmo es un conjunto de esquemas MER.

1.- Encontrar las entidades con atributos numéricos y crear un nodo de hechos por cada una de ellas.

En este paso, se identifican las entidades con atributos numéricos que serán consideradas hechos. En el ejemplo, las entidades del diagrama ER identificadas como hechos son: Línea (2), Ticket (1), Artículo (1), una vez identificadas se les concatena la etiquete "*event*" a cada una de ellas lo que indica que es un hecho, el símbolo usado para representar un hecho en MER es un diamante como se muestra en la figura 4.16.

Figura 4.16. Representación de un hecho en MER

2.- En el paso 2, los atributos numéricos de las entidades del esquema ER consideradas hechos son agregados al diamante, ver figura 4.17.

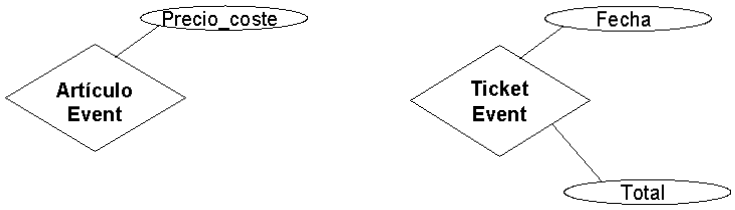


Figura 4.17. Representación de un hecho

3.- En este paso, se identifican los atributos tipo fecha de las entidades elegidas como hechos.

Estos atributos formaran la dimensión fecha del esquema MER. En este ejemplo, la entidad Ticket es la única entidad con un atributo tipo fecha por lo que este atributo es considerado una dimensión.

Debido a que la granularidad de la fecha es desconocida en este punto de diseño esta dimensión deberá ser refinada en base a los requisitos de usuario, en MER las dimensiones que necesitan ser refinadas se representan por medio de un hexágono. En la figura 4.18 se muestra que la dimensión fecha debe ser refinada en base a los requisitos de usuario.

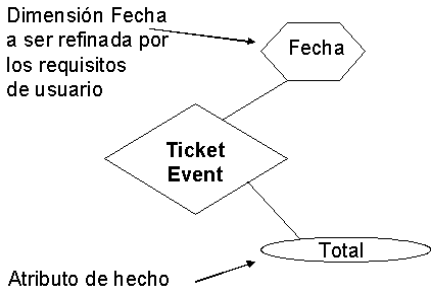


Figura 4.18. Representación de la dimensión Tiempo

4.- El paso 4 se realiza solamente si las entidades consideradas como hechos tienen atributos textuales.

Si existe un atributo textual en alguna de ellas, se debe crear un nivel asociado al hecho el cual representara una dimensión. En la figura 4.19 se observa que el hecho *Artículo Event*, tiene asociada la dimensión Artículo, la cual a su vez esta formada por los atributos textuales (descripción). El nombre de la nueva dimensión será el nombre de la entidad identificada como hecho en el esquema ER y el símbolo usado para representarla en el esquema multidimensional es un rectángulo.

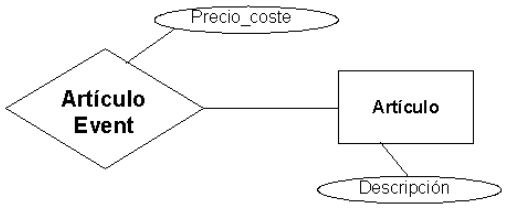


Figura 4.19. Dimensión asociada al hecho Artículo

5.- El paso 5, consiste en hacer un recorrido recursivo al diagrama ER a partir de las entidades identificadas como hechos, la condición para agregar una entidad al esquema multidimensional es que la relación entre las entidades sea Muchos a Uno o Muchos a Muchos. En la figura 4.20, se muestra el esquema multidimensional para el hecho Ticket el cual se obtuvo al recorrer el esquema ER. El esquema multidimensional para Artículos se muestra en la figura 4.21. Debido a que en el esquema multidimensional de Artículo (figura 4.21), no aparece la dimensión fecha y que en todo esquema multidimensional debe aparecer, es necesario agregar esta dimensión en el esquema

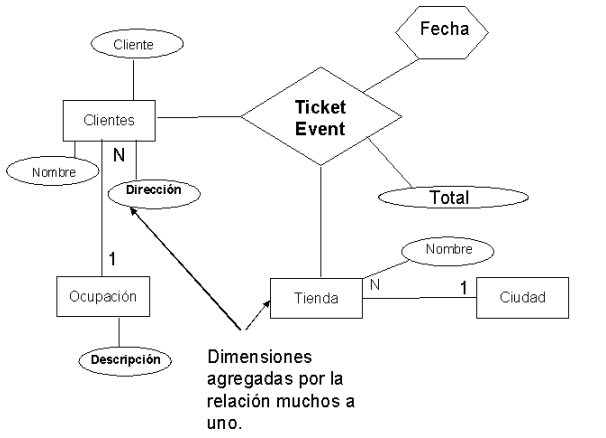


Figura 4.20. Esquema multidimensional de Ticket

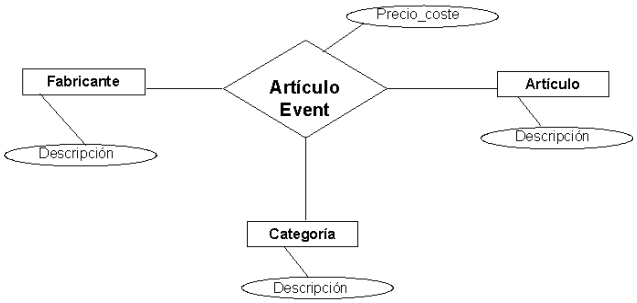


Figura 4.21. Esquema multidimensional de Artículo

El símbolo utilizado para esto es una nube la cual indica que la dimensión no fue derivada automáticamente del esquema ER y que requiere ser refinada a partir de los requisitos de usuario (figura 4.22). El esquema multidimensional para Línea se muestra en la figura 4.23.

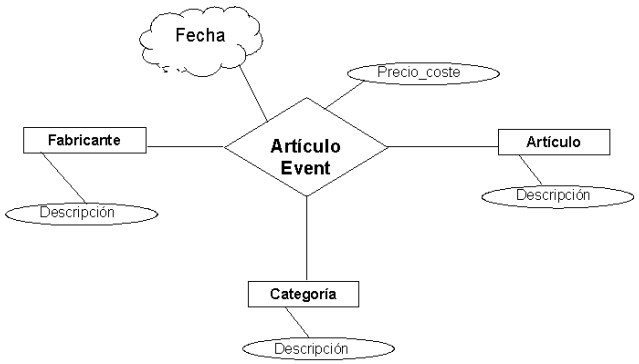


Figura 4.22. Nube que representa una dimensión agregada

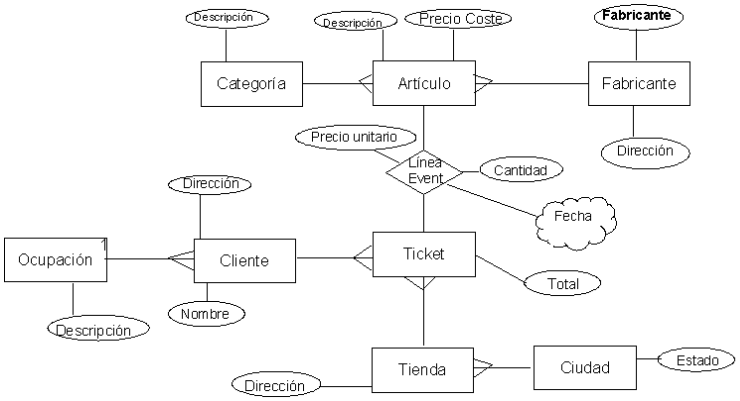


Figura 4.23. Esquema multidimensional de Línea

Selección del esquema candidato y refinamiento: Una vez identificadas las dimensiones se hace un refinamiento manual de los esquemas multidimensionales.

El refinamiento se hace en base a las consultas de usuario, el objetivo de esta etapa es seleccionar los esquemas que den soporte a las aplicaciones de usuarios.

Hay dos aspectos de las consultas que son usados para determinar si un esquema multidimensional puede dar respuesta a la consulta, las tablas en la cláusula FROM y los campos numéricos en la cláusula SELECT.

Si un esquema candidato no contiene la tabla(s) en la cláusula FROM este no puede dar respuesta a al consulta por lo que el esquema se descarta, de lo contrario es necesario analizar si aparecen los campos de la cláusula SELECT.

En esta etapa el autor recomienda hacer una tabla de comparación entre esquemas y consultas. En la tabla se registra por medio de una "P" si una consulta puede ser resuelta parcialmente por el esquema, una "X" significa que el esquema da una respuesta total y un espacio en blanco que el esquema no da respuesta a la consulta.

La consulta de la figura 4.24(a) muestra los 10 artículos que tuvieron menos ventas durante el año 2000 y la consulta de la figura 4.23(b) muestra las cinco categorías de artículos que tuvieron mayor promedio de ventas durante el año 2000.

En la tabla de comparación de la figura 4.25, se destaca que el esquema que cumple con las dos consultas es Línea Event.

```
SELECT TOP 5  AVG(Total) AS total, Categoría.descripción
FROM Línea event, Artículo, Categoría, Fecha
WHERE línea Event.id_artículo=Artículo.id_artículo AND
        Artículo.id_categoria=Categoría.id_categoría AND
        Event.id_fecha=Fecha.id_fecha AND Fecha.año=2000
GROUP BY Categoría.descripción
```

b)Relación de artículos con mayor promedio de ventas

```
SELECT TOP 10 SUM(Cantidad) AS Total, Artículo.descripción
FROM Línea Event, Artículo, Fecha
WHERE  Línea Event.id_articulo=Artículo.id_artículo AND
       Línea Event.id_fecha= Fecha.id_fecha AND
       Fecha.año=2000
GROUP BY Artículo.descripción
ORDER BY  SUM(Cantidad) DESC
```

a)Relación de artículos con menos ventas

Figura 4.24. Consultas de usuario

Esquema / Consulta	Artículo Event	Ticket Event	Línea Event
a)			X
b)			X

Figura 4.25. Tabla de comparación entre esquemas y consultas

Por último, es necesario hacer un refinamiento manual del esquema multidimensional. En esta etapa se decide entre otras cosas la granularidad de las dimensiones representadas con un hexágono o con una nube. Por ejemplo si las consultas de usuario son por semana o anuales es necesario representar estos niveles en la dimensión Fecha.

En resumen este modelo permite representar los conceptos básicos del modelado multidimensional como son los hechos, las dimensiones, las medidas, las relaciones disjuntas, más no permite representar las relaciones Muchos a Muchos entre los hechos y las dimensiones, la relación de completitud, las jerarquías cubiertas, la categorización de dimensiones y no permite representar propiedades de aditividad.

Además, es importante resaltar que al considerar sólo los atributos numéricos para identificar hechos el algoritmo excluye las relaciones Muchos a Muchos sin atributos numéricos las cuales representan un *Fact Less table* en el modelo lógico.

4.1.5. Resumen de propiedades.

En la figura 4.26 se muestra un resumen de las propiedades que pueden representarse en los modelos de datos revisados. Sólo DFM permite representar propiedades de aditividad y la categorización de dimensiones, ninguno de los modelos permite representar las jerarquías completas y las jerarquías cubiertas, sólo MER permite representar las relaciones Muchos a Muchos y las jerarquías disjuntas.

Modelo / Propiedad	DFM	MD	ME/R
Aditividad	√	χ	χ
Jerarquías disjuntas	χ	χ	√
Jerarquías completas	χ	χ	χ
Jerarquías cubiertas	χ	χ	χ
Relaciones Muchos a Muchos entre hechos y dim.	χ	χ	√
Categorización de dim.	√	χ	χ

Figura 4.26. Propiedades de modelado multidimensional soportadas por cada modelo de datos

4.2. Metodologías dirigidas por los requisitos de usuarios.

En esta sección, se presentan un análisis de dos metodologías de diseño de AD que derivan el esquema del AD a partir de los requisitos de usuario.

4.2.1. Metodología para la elicitación de requisitos basada en I*.

En [MTSP05], se propone una metodología para la elicitación de requisitos que se compone por tres fases:

1. Identificación de los objetivos estratégicos.- Los objetivos estratégicos, describen las metas que la organización pretende lograr con la participación de ciertos actores. La definición de los objetivos estratégicos, inicia a partir de las metas que los actores se plantean. Una vez definidos los objetivos estratégicos, se forma una jerarquía de objetivos de más bajo nivel siguiendo la siguiente dependencia: objetivos de decisión, objetivos de información y requisitos de información.

    - Objetivos de decisión.- Estos objetivos, representan un nivel de abstracción medio y responden a la pregunta: cómo puede un objetivo estratégico ser resuelto?.
    - Objetivos de información.- Estos objetivos, representan un nivel de abstracción más bajo en la dependencia definida y responden a la pregunta: cómo puede un objetivo de decisión ser resuelto?.

- Requisitos de información.- Estos requisitos, se obtienen a partir de los objetivos de decisión y de los objetivos de información.

2. Construcción del modelo estratégico y de dependencias empleando el marco de trabajo i*.- Con la finalidad de representar gráficamente la relación entre los actores y los distintos objetivos, se utiliza la técnica i*. Esta técnica, esta formada por el modelo de dependencias estratégicas (DE) y el modelo de razones estratégicas (RE) los cuales, se complementan entres si para proporcionar una visión general sobre de la naturaleza de la organización.

   Para la construcción del modelo de DE se proponen tres pasos: 1) Identificar los actores del negocio, 2) Determinar los objetivos estratégicos, 3) Descubrir los requisitos de información. Para la construcción del modelo de RE, se propone una guía compuesta por dos puntos: 1) Identificación de tareas y metas, 2) Representación de las tareas.
3. Construcción del modelo multidimensional.- Una vez definido el modelo DE y el modelo RE, los autores proponen la construcción del modelo multidimensional relacionando conceptos de los modelos DE y RE con el modelo multidimensional.

Caso de estudio (Ejemplo).- La primera fase de la metodología, consiste en identificar los objetivos estratégicos a partir de los actores que participan en la toma de decisiones. En nuestro caso de estudio, supondremos que el actor identificado es: *gestor de ventas*.

El objetivo estratégico que se puede identificar al interactuar con el actor es: *incrementar el número de clientes*. Los objetivos decisionales se obtienen al responder a la pregunta: ¿*Cómo se puede incrementar el número de clientes*?. R: *estableciendo promociones efectivas.*

Los objetivos de información, se obtienen al responder a la pregunta: ¿Cómo puede determinarse una promoción efectiva?. R: analizando promociones previas y comportamientos en las compras de los clientes.

Los requisitos de información, se obtienen responder a la pregunta: ¿Cómo puedo analizar el comportamiento de los clientes? R: Analizando la cantidad vendida durante las promociones.

En la figura 4.27, se muestra que el *gestor de ventas* se interesa en el estudio de la información acerca de las ventas, específicamente en el análisis de la información que permita incrementar el número de clientes. En el modelo de RE, se representan la relación entre el actor y los diferentes tipos de objetivos identificados (figura 4.28).

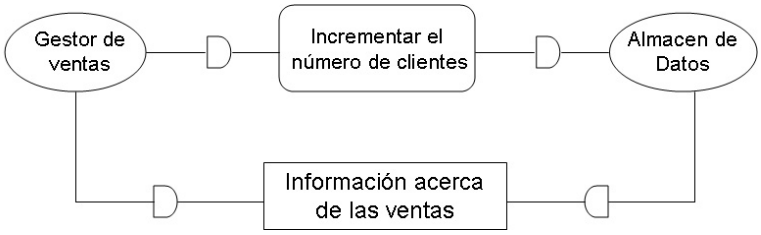


Figura 4.27. Modelo de dependencia estratégica

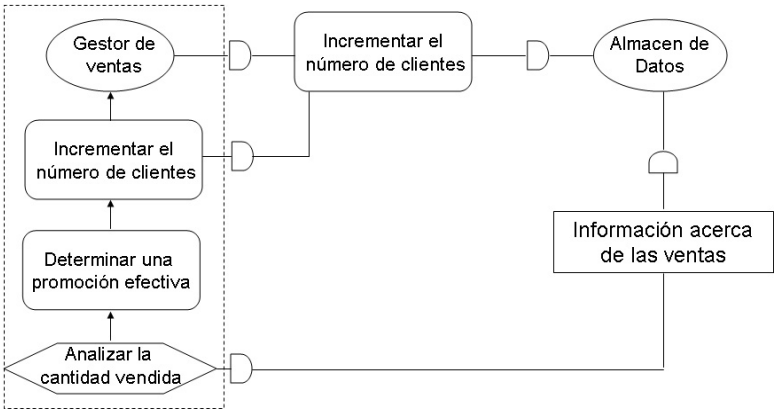


Figura 4.28. Modelo de razones estratégicas

Una vez identificados los objetivos, el autor propone que se realice un análisis dentro del contexto del problema para identificar las tareas que se relacionan con los requisitos de información.

Por ejemplo, las tareas relacionadas con el requisito de información *analizar la cantidad vendida* son: analizar la cantidad vendida por producto, analizar la cantidad vendida por fecha, analizar la cantidad vendida por tienda (figura 4.29).

A partir de la información representada en el modelo de razones estratégicas, se identifica la clase de hechos *Ventas.* El atributo que se incluirá en la clase de hechos *Ventas* es *cantidad vendida*. Las dimensiones, son identificadas a partir de la información relacionada con el contexto de análisis, en este caso: *fecha, producto y tienda*.

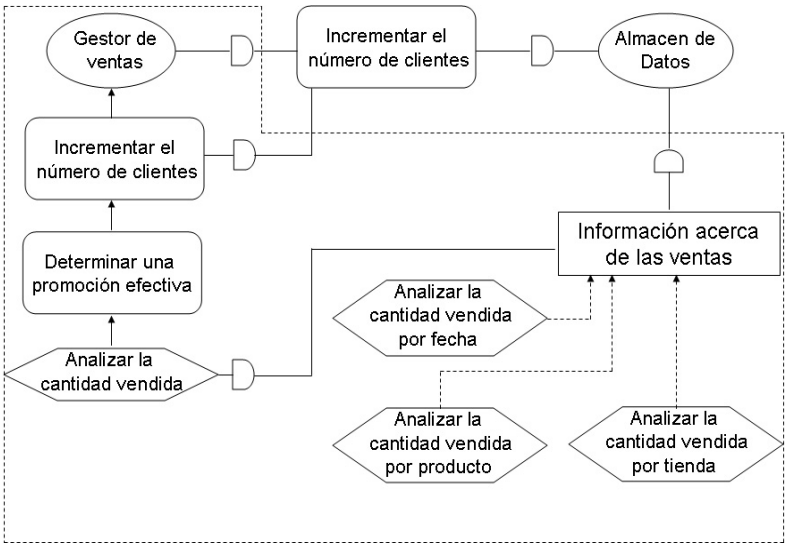


Figura 4.29. Modelo de razones estrategias y tareas

4.2.2. Modelo Wal-Mart.

En [Wes01] se presenta el modelo Wal-Mart, el cual inicia la construcción de ADs a partir de los requisitos de usuario. La parte principal de esta propuesta, la constituye la exploración del proceso de negocio que es objeto de estudio.

El método que se propone se forma por cuatro pasos:

1. Definir las metas.- En este paso, se delimitan las metas a cumplir con la construcción del AD. Por ejemplo: "*Utilizar los datos existentes para proporcionar información que permita definir estrategias de comercialización de productos*"

2. Obtener las preguntas del negocio**.-** Considerando que el AD debe proporcionar información que permita responder a las preguntas del negocio. En este paso, se definen las preguntas que el AD deberá responder. Por ejemplo: *"Cuál es el margen de ganancias de un artículo por ciudad?".*

3. Definir la prioridad para las preguntas del negocio**.-** Este paso, consiste en ordenar las preguntas identificas en relación a su importancia. Para lo cual, las preguntas, se organizan en una forma denominada "*Forma de prioridades*". Ésta, se compone de cuatro columnas:

   - Meta.- **E**n esta columna, se indica si la pregunta se relaciona directamente con la meta.

   - Grupo.- En esta columna, se relaciona cada pregunta con la meta en base a su importancia, para lo cual define un rango de valores del 1 al 5.

   - Prioridad.- En esta columna, se define la precedencia de cada pregunta dentro de la meta, para lo cual utiliza un rango de valores del 1 al 5.

   - Pregunta.- En esta columna, se describen las preguntas.

Por ejemplo, en la tabla 1, se muestra que la pregunta "*Qué tan grande debe ser una tienda?*", no se relaciona con la meta, por lo que las columnas grupo e importancia tienen el valor 5 (sin importancia).

Tabla 4.1 Forma de prioridades

Meta	Grupo	Prioridad	Pregunta
Si	1	1	Qué porcentaje de contribución tiene un articulo con el margen de ganancia total ?
Si	1	1	Qué tiendas presentan el margen de ganancias mayor?
Si	2	2	Cuáles son las ventas diarias de un artículo?
No	5	5	Qué tan grande debe ser una tienda?

4. Definir las preguntas del negocio**. -** El propósito de este paso, es definir los datos necesarios para responder a las preguntas del negocio. Específicamente, es necesario documentar las jerarquías entre los datos identificados y las métricas necesarias para dichos datos.

   El resultado de éste paso, es un glosario que contiene la definición de las jerarquías y las métricas. Por ejemplo:

   Las métricas para artículos pueden ser: *ventas, unidades vendidas, etc*. La jerarquía puede ser: *departamento, clase, artículo*.

4.2.3. Resumen de propiedades.

El inconveniente principal de las metodologías previamente analizadas, es el débil acoplamiento que existe entre los esquemas multidimensionales obtenidos y las bases de datos operacionales.

Este débil acoplamiento, dificulta el proceso de extracción, transformación y carga. Además, es difícil garantizar que los datos requeridos por los usuarios se encuentran disponibles en las bases de datos operacionales. Una carencia adicional

de las propuestas analizadas es la notación utilizada para identificar los requisitos de usuario.

Ambas propuestas, utilizan una notación no estandarizada que dificulta la comprensión de los requisitos por parte de los usuarios y los desarrolladores del sistema. También, se dificulta la realización de los cambios en los requisitos de manera fácil y consistente. Para representar el esquema conceptual del AD, en [MTSP05] los autores proponen el perfil UML presentado en [LTS02a] [LTS02b]. En [Wes01] no se especifica ninguna notación.

4.3. Metodologías compuestas.

En esta sección, se presentan un análisis de dos metodologías de diseño de AD que derivan el esquema del AD a partir de los requisitos de usuario y del análisis de la base de datos operacional.

4.3.1. Goal-Oriented Requirement Analysis for Data Warehouse Design.

En [GRG05] se propone una metodología que se fundamenta en Tropos [BGGMP04]. Esta metodología, puede ser empleada de dos formas: como técnica de diseño dirigida por los requisitos de usuario o bien como técnica de diseño compuesta.

Análisis de requisitos.- Como se mencionó anteriormente esta metodología se fundamenta en Tropos. Tropos, es una metodología de desarrollo de software basado en el marco de trabajo I*.

En este marco de trabajo, el concepto de agente y meta son usados para el desarrollo del AD. Durante las primeras etapas del análisis de requisitos, se deben identificar a los agentes que se encuentran dentro del dominio del problema y considerarlos actores del sistema.

Estos actores, dependen de alguna meta a ser realizada. Los conceptos de Tropos que se utilizan en el contexto de ADs son:

- Actor. Un actor, representa un agente que puede ser modelado y que juega un papel importante en la toma de decisiones
- Dependencias. Una dependencia, representa un *agrigament* entre dos actores.

  El *agrigament* puede ser una meta o una tarea a ser realizada o bien un recurso.
- Diagrama de actores. Es un grafo donde los actores se relacionan entre si por medio de dependencias.

  Este grafo, es usado para modelar tales dependencias.
- Diagrama de razones. Este diagrama, es usado para representar el fundamento lógico de las relaciones entre los actores.

  En él, las metas se representan como círculos con los que se relaciona un actor.

  Las metas, son descompuestas en sub-metas por medio de relaciones semánticas *And/Or*.

Una semántica *And* significa que todas las sub-metas deben realizarse mientras que una semántica *Or* se interpreta como cualquiera de las sub-metas debe realizarse.

Cuando se realiza el análisis de los requisitos de usuario, deben considerarse dos perspectivas.

Primero, debe analizarse la organización en la cual el AD operará (modelo organizacional).

Segundo, con el objetivo de identificar los requisitos funcionales y no funcionales del AD, es necesario diseñar un conjunto de diagramas de razones (modelo de decisiones)

Modelo organizacional.- El modelo organizacional, se compone por tres fases:

1. Análisis de metas**.-** Donde un actor y un diagrama de razones son relacionados entre si.
2. Análisis de hechos.**-** En el cual, un diagrama de razones es extendido con hechos.
3. Análisis de atributos**.-** Este análisis, permite extender el diagrama de razones al agregarle atributos.

Cada fase, es un proceso iterativo que inicia a partir del diagrama producido en la fase anterior.

1. Análisis de metas. El primer paso, consiste en identificar a los actores y sus dependencias, lo cual se realiza por medio de un diagrama de actores. La figura 4.30, muestra un diagrama de actores En el cual, el *gestor de ventas* depende del AD para lograr

la meta *manejar ventas*. El siguiente paso, consiste en analizar las metas de cada actor con más detalle para producir un diagrama de razones. Las metas, son divididas por medio de relaciones semánticas *AND*.

Figura 4.30. Diagrama de actores

En la figura 4.31, se muestra el diagrama de razones para el actor *Gestor de Ventas*. En la figura 4.31, se muestra que la meta manejar ventas se ha dividido en dos sub-metas: *Manejar ventas con promociones* y *Manejar ventas sin promociones*

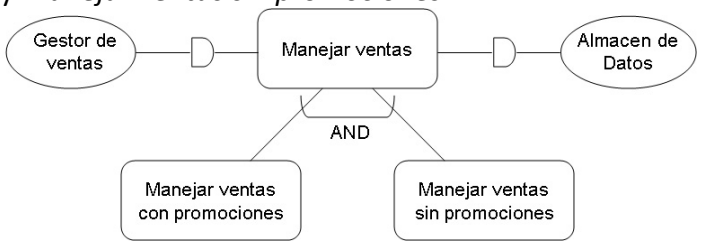


Figura 4.31. Diagrama de razones

2. Análisis de hechos. El objetivo de esta fase, consiste en identificar los hechos más relevantes de la organización. Este proceso, se lleva a cabo navegando por el diagrama de razones y extendiéndolo por medio de asociaciones entre las metas y los hechos.

La figura 4.32, muestra el diagrama de razones extendido. El hecho *Promociones* es asociado a la meta *Manejar ventas con promociones.*

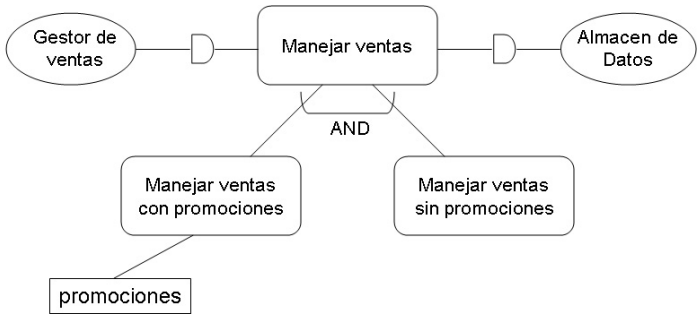


Figura 4.32. Diagrama de razones extendido

3. Análisis de atributos. Esta fase, esta dedicado a identificar y asociar los atributos que pueden dar un valor a los hechos. Este análisis, parte del modelo de razones extendido y asocia atributos a las sub-metas identificadas. Durante esta fase, los atributos son identificados sin catalogarlos como medidas o dimensiones. La figura 4.33, muestra el diagrama de razones extendido al cual se le han agregado los atributos.

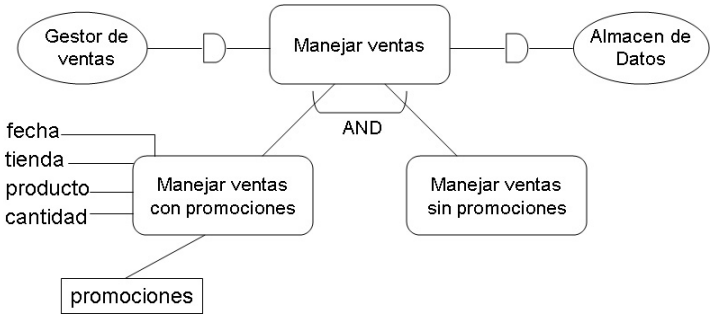


Figura 4.33. Diagrama de razones extendido con atributos

Modelo de decisiones.- Este modelo, tiene como objetivo analizar con más detalle las metas. Por cada actor identificado se deben realizar los siguientes pasos:

1. Análisis de metas. Este paso, consiste en analizar las metas de cada actor con más detalle con el objetivo de producir un nuevo diagrama de razones.

   Las metas identificadas en este paso, pueden ser completamente diferentes a las identificadas en el modelo organizacional y estas deben ser divididas por medio de relaciones semánticas *ORs*.

   En la figura 4.34, se muestra el diagrama de razones para el análisis de la meta *analizar promociones*.

   El cual, puede llevarse a cabo por medio del análisis de las *promociones a crédito* o bien por el análisis de las *promociones al contado*.

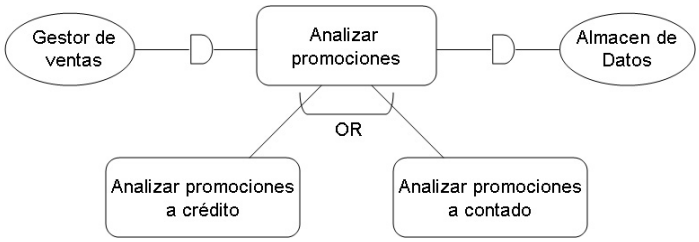


Figura 4.34. Diagrama de razones con relaciones OR

2. Análisis de los hechos. Al igual que en el modelo organizacional, el diagrama de razones es extendido al agregarle los hechos a las metas.

   Los hechos son importados del diagrama de razones producido en el modelo organizacional. En la figura 4.35, se introdujo el hecho *promociones* en el diagrama de la figura 4.34.

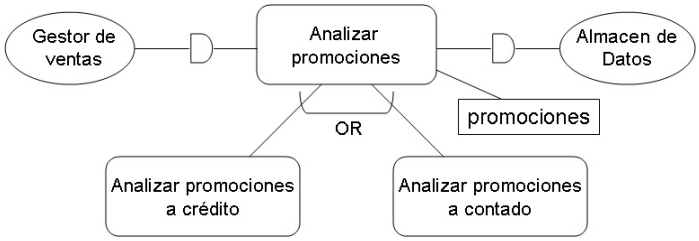


Figura 4.35. Diagrama de razones con hecho asociado

3. Análisis de dimensiones y análisis de medidas. En esta fase, cada hecho se relaciona con las dimensiones y medidas.

   Las dimensiones y medidas se identifican al interactuar con el actor del diagrama de razones (figura 4.36).

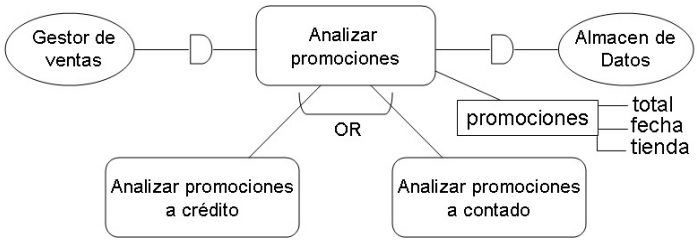


Figura 4.36. Diagrama de razones con dimensiones y medidas

Marco de trabajo para el diseño del AD compuesto.- En esta fase, el autor proporciona una guía para relacionar la información obtenida durante la fase de requisitos con el esquema de la base de datos operacional.

Esta guía, se compone de tres pasos:

1*) Relación de requisitos*.- En este paso, los hechos, las dimensiones y las medidas que se encuentran en el diagrama de razones se relacionan con el esquema de la base de datos operacional.

Para esto, el autor propone que los hechos del modelo de decisiones se relacionen con entidades *N-arias*.

Las dimensiones y medidas se identifican relacionando los atributos del modelo organizacional con los atributos del esquema fuente.

En nuestro ejemplo, el hecho *Promociones* del diagrama de razones (figura 4.36) puede relacionarse con la entidad *Promociones* de algún esquema fuente.

De igual forma, es necesario comparar los atributos del hecho *Promociones* con los atributos de las tablas del esquema operacional.

2). *Construcción de jerarquías*.- En este paso, el esquema conceptual del AD se construye navegando en el esquema de la base de datos operacional.

Para lo cual, el autor propone que se realice un recorrido a través de las asociaciones muchos a uno del esquema de la

base de datos operacional a partir de la entidad identificada como hecho.

La notación que el autor utiliza para representar el esquema conceptual del AD, se presenta en [GMR98a]

3) *Refinamiento*.- En este paso, el esquema conceptual del AD es editado manualmente para cumplir las expectativas de los usuarios.

Los pasos que el autor propone son similares a los propuestos en [GMR98a].

4.3.2. Designing Data Marts for Data Warehouses.

En [BCCFP01], se presenta una metodología para la construcción de *data marts* a partir del esquema conceptual de la base de datos operacional y de los requisitos de usuario.

El método que propone, consiste de tres fases:

1. En esta fase, los requisitos de usuario se identifican por medio de entrevistas. La información generada por las entrevistas se concentra en una forma utilizando el método *Goal/Question/Metrics (GQM)*.
2. La segunda fase, esta dedicada a examinar el esquema conceptual de la base de datos operacional. El objetivo de esta fase, es generar un conjunto de *data marts* empleando una técnica de análisis basada en grafos.

3. En la tercera fase se selecciona el *data mart* final.

A continuación, se explica la metodología por medio de un ejemplo:

1.- Definición de los requisitos de usuario.- La identificación de los requisitos de usuario, se realiza por medio del método GQM.

Este método, utiliza dos formas de abstracción en las que se resumen las características de los procesos del negocio.

La primera forma, se usa para definir un objetivo organizacional en términos abstractos, en ella se define el objeto de estudio, los factores de calidad, los puntos de vista y el ambiente.

El objeto de estudio se refiere al proceso de negocio que se analiza. Los factores de calidad, reflejan los aspectos del proceso de negocio que es objeto de estudio.

El punto de vista, se refiere al usuario. Finalmente, el ambiente se refiere al departamento donde se utilizará el *data mart*.

La segunda forma, se usa para enriquecer la definición del objetivo organizacional. En la figura 4.37, se muestra un ejemplo con las dos formas de abstracción.

La información contenida en las formas, es interpretada de la siguiente manera: la columna factores de calidad, representan los atributos que contendrá el *data mart*, mientras que los factores de variación, representan las dimensiones.

Así, *promociones, tienda, ofertas y tipo de artículo* son considerados dimensiones mientras que *ventas* es considerado atributo de un data mart.

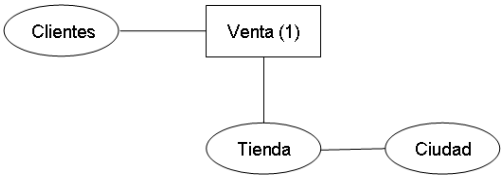
Objeto de estudio	Propósito	Factores de calidad	Puntos de vista	Ambiente
Proceso de ventas	Por definir	Ventas	Gerente de ventas	Actividades desarrolladas por el departamento de ventas

Factores de calidad	Factores de variación
Ventas	
Ventas por tienda	Promociones, Ofertas, tienda
Ventas por artículo	Tipo de artículo

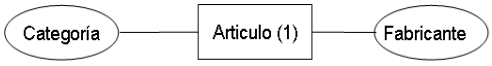
Figura 4.37. Forma de abstracción GQM

2.- Obtención de los data marts.- En esta fase, se realiza un análisis del esquema ER con la finalidad de generar un conjunto de grafos, donde cada grafo será considerado un data mart.

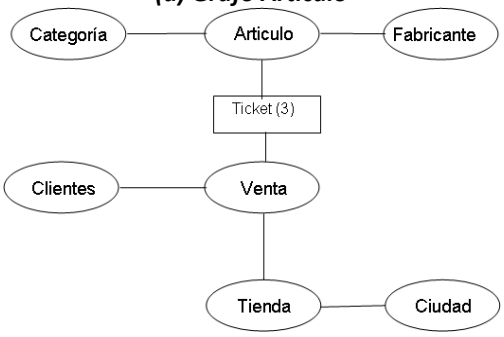
Los grafos, son generados por un algoritmo que realiza un recorrido en profundidad en el esquema ER. El algoritmo identifica las entidades con atributos numéricos y las etiqueta con el número de atributos numéricos que las compone. Estas entidades son consideradas entidades de hechos. Las dimensiones se identifican a partir de cada entidad de hecho realizada un recorrido en el esquema ER a través de las relaciones muchos a uno, Al aplicar el algoritmo al esquema conceptual de la figura 4.1, se generan 3 grafos (figura 4.38).



(b) Grafo Venta



(a) Grafo Artículo



(c) Grafo Ticket

Figura 4.38. Grafos que representan esquemas multidimensionales

La fase 2, finaliza con un resumen sobre las propiedades de cada grafo. En la Tabla 4.2, se muestra que el grafo Ticket tiene siete dimensiones y tres atributos numéricos. Esta información será utilizada por la siguiente fase para elegir un grafo.

3.-Selección.- Esta fase, consiste en relacionar los requisitos de usuario con los grafos generados en la fase anterior.

Lo cual, se realiza en tres pasos: análisis de terminología, comparación de esquemas y selección.

Tabla 4.2. Resumen de propiedades

Grafo	Número de dimensiones	Número de atributos numéricos
Ticket	7	3
Articulo	2	1
Venta	3	1

El análisis de terminología, consiste en identificar las relaciones semánticas existentes entre los elementos de cada grafo y los requisitos de usuario para lo cual, es necesario interactuar con los usuarios. En nuestro ejemplo, se observa que el elemento *Venta* del grafo *Venta* se relaciona directamente con el factor de variación *Ventas* de la forma de abstracción.

Una vez obtenido el conjunto de relaciones semánticas, se selecciona el grafo.

4.3.3. Resumen de propiedades.

Un inconveniente de las propuestas revisadas, es la notación propietaria que utilizan para representar el esquema conceptual del AD. El uso de una notación propietaria y el método utilizado para identificar los requisitos de usuario (no estandarizado), convierten a estas propuestas en metodologías aisladas y difíciles de utilizar.

4.4. Análisis comparativo.

En este capítulo, hemos presentado un breve análisis de las principales propuestas relacionadas con el diseño conceptual de ADs. En este análisis nos hemos centrado en la estrategia propuesta por los autores para obtener el esquema conceptual de la base de datos del AD.

De esta forma, hemos presentado primero aquellas investigaciones que inician el diseño del AD a partir de la descripción conceptual de la base de datos operacional (sección 4.1). Posteriormente, en la sección 4.2, hemos descrito las propuestas que parten de los requisitos de usuario, pero que ignoraran la estructura de las bases de datos operacionales. Finalmente, en la sección 4.3, hemos presentado las propuestas compuestas.

A continuación presentamos las conclusiones derivadas del análisis realizado.

Propuestas dirigidas por datos: las propuestas presentadas [GMR98a], [GMR98b], [CT98], [PD02], pertenecen al grupo de metodologías que no consideran los requisitos de usuario para el diseño conceptual del AD. Todas estas propuestas, parten del análisis del esquema ER de la base de datos operacional.

Las metodologías analizadas, proporcionan algoritmos para identificar en el esquema ER, los elementos que se corresponden semánticamente con los conceptos del modelado multidimensional.

Propuestas dirigidas por procesos: el inconveniente principal de las propuestas dirigidas por los requisitos de usuario [Wes01],[MTSP05], es el débil acoplamiento de los esquemas multidimensionales resultantes con las bases de datos operacionales.

Este débil acoplamiento, dificulta el proceso de extracción, transformación y carga. Además, no se garantiza que los datos solicitados por los usuarios se encuentran disponibles en las bases de datos operacionales.

Propuestas compuestas: la principal debilidad de las propuestas analizadas [GRG05], [BCCFP01], es la notación utilizada para identificar los requisitos de usuario. Ambas propuestas, utilizan una notación no estandarizada que dificulta la comprensión de los requisitos por parte de los usuarios y los desarrolladores del sistema.

También, se dificulta la realización de los cambios en los requisitos de manera fácil y consistente. El uso de una notación propietaria y el método utilizado para identificar los requisitos de usuario (no estandarizado), convierten a estas propuestas en metodologías aisladas y difíciles de utilizar.

Para concluir esta sección, podemos resumir lo siguiente:

1. Las propuestas que siguen la aproximación de analizar el esquema conceptual de la base de datos operacional usan algoritmos particulares para explorar el esquema conceptual y generar el esquema multidimensional del AD. En este sentido, nuestro esfuerzo ha consistido en utilizar un

mecanismo estandarizado (basado en MDA), para realizar este proceso.

2. Las propuestas que siguen la aproximación de analizar los requisitos de usuario, utilizan mecanismos que dificultan la comunicación de estos requisitos, y las guías que proporcionan para relacionar los requisitos con los conceptos del modelado multidimensional no son claras. En este sentido, proponemos el uso de casos de uso y diagramas de actividad (basadas en UML).

3. Por último se observa, que se han definido pocas técnicas para el diseño conceptual de ADs que integren los requisitos de usuarios con el análisis del esquema conceptual de la base de datos operacional. En este trabajo se propone un conjunto de reglas que permiten relacionar los requisitos de usuario con los esquemas multidimensionales. Aunque este paso es considerado en algunas propuestas analizadas, hemos extendido el conjunto de reglas de tal forma que la selección sea más precisa.

Capítulo 5

Una metodología de diseño conceptual de almacenes de datos.

En este capítulo, se propone una metodología para el diseño conceptual de almacenes de datos. La metodología consta de tres fases: 1) derivación de esquemas multidimensionales, 2) análisis de requisitos de usuario, 3) integración. La metodología que se propone se basa en los estándares que integran MDA y en las herramientas que se han desarrollado en torno a este estándar.

5.1. Diseño conceptual de almacenes de datos.

Como es bien sabido, los métodos clásicos de diseño de bases de datos operacionales se estructuran en una secuencia de etapas, que se inicia con el análisis de los requisitos de usuario para a partir de ellos obtener el esquema conceptual, a continuación el esquema lógico y finalmente el esquema físico de la BD.

Un AD, es una base de datos con características de volumen y explotación distintas a las bases de datos operacionales. Una base de datos operacional se diseña para soportar los procesos básicos de la organización mientras que un AD se diseña para hacer análisis de datos. Esta diferencia de uso, modifica significativamente las aproximaciones al diseño de ADs.

El problema básico del diseño de un AD consiste en obtener un conjunto de esquemas multidimensionales que permitan satisfacer los requisitos de análisis de los usuarios y que puedan ser mantenidos por las bases de datos operacionales existentes en la organización. La figura 5.1, muestra las etapas del diseño de un AD.

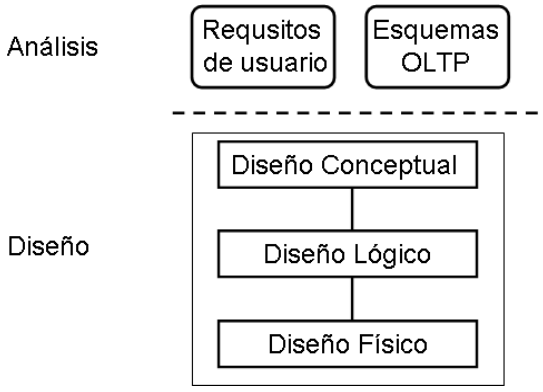


Figura 5.1. Etapas para el diseño de un AD

El objetivo de este trabajo es desarrollar una metodología para el diseño conceptual de ADs a parir de la descripción conceptual de las bases de datos operacionales y de los requisitos de usuario, para lo cual se definen tres fases:

- Fase 1. Derivación de esquemas multidimensionales: consiste en la obtención de un conjunto de esquemas multidimensionales candidatos a partir del esquema conceptual de la base de datos operacional. Se asume que el esquema conceptual disponible está en el modelo Entidad Relación (ER).
- Fase 2. Especificación de requisitos de usuario: consiste en identificar las necesidades de análisis de los usuarios.
- Fase 3. Integración: consiste en combinar el conocimiento obtenido en las dos fases anteriores.

La primera fase, está dedicada a examinar el esquema conceptual de la base de datos operacional, para generar los esquemas multidimensionales candidatos para el AD. El

desarrollo de esta fase, se ha abordado en el contexto de MDA, para ello hemos definido un conjunto de reglas de transformación entre el metamodelo ER y el metamodelo OLAP.

En la segunda fase, los requisitos de usuario son recogidos por medio de entrevistas. El propósito de las entrevistas es obtener información acerca de las necesidades de análisis de la organización. En esta fase, adaptamos un método de elicitación de requisitos basado en metas.

En la tercera fase, se contrasta la información obtenida en la segunda fase con los esquemas multidimensionales candidatos, generados en la primera, con el objetivo de seleccionar el esquema multidimensional candidato más adecuado. Posteriormente, será responsabilidad del diseñador refinar manualmente el esquema multidimensional seleccionado.

El método que proponemos, integra dos perspectivas de diseño de ADs complementarias. El análisis del esquema ER que se realiza en la fase 1, muestra los esquemas multidimensionales implícitos en la base de datos operacional, mientras que en la fase 2, se enfatizan las necesidades de los usuarios.

La fase final integra estos dos puntos de vista y genera así, un esquema multidimensional que contempla los requisitos de usuario y puede ser soportada por las bases de datos operacionales. Las características principales de la metodología son:

- La obtención de los esquemas candidatos del AD se realiza de manera automática por medio de un conjunto de reglas de transformación definidas en el contexto de MDA.
- Se emplea un método basado en metas para la elicitación de requisitos.

5.2. Descripción de metamodelos.

Common Warehouse Metamodel (CWM), es un estándar definido específicamente para modelar aplicaciones de bases de datos. Este estándar, tiene metamodelos específicos para el modelado de bases de datos relacionales, OLAP, XML, ER, etc. Aunque CWM fue definido como un estándar para el intercambio de datos entre aplicaciones y herramientas, nosotros utilizamos los metamodelos ER y OLAP de este estándar para la definición del esquema conceptual de la base de datos operacional y del AD respectivamente.

5.2.1. Metamodelo Entidad Relación.

El modelo[5] ER fue propuesto en 1976 por Peter Chen [CPP76], desde su introducción, este modelo de datos ha gozado de gran aceptación en el ámbito de las bases de datos. El modelo ER es considerado el precursor de los

[5]El modelo Entidad Relación (ER) es un modelo de datos semántico utilizado para el diseño conceptual de bases de datos. En la terminología de MDA lo que tradicionalmente se refiere como modelo ER, es el metamodelo ER, y los esquemas conceptuales definidos con ER (diagramas ER) son lo que en el marco de MDA denominamos modelos ER. Ya que el trabajo se ha desarrollado en el marco de MDA, en este capítulo se va a intentar seguir su terminología: metamodelo-modelo.

modelos actuales y es probablemente el primer modelo de datos que tiene una semántica asociada a él.

Este modelo de datos se utiliza para describir la estructura de una base de datos en el nivel conceptual. Debido a la importancia que el modelo ER ha adquirido, CWM incluye como parte de su especificación el metamodelo ER [OMG03b]. El metamodelo ER se define por un conjunto de clases derivadas del metamodelo de UML. Como se puede ver en la figura 5.2 muchos de los conceptos del metamodelo ER tienen una equivalencia directa con las clases del metamodelo UML.

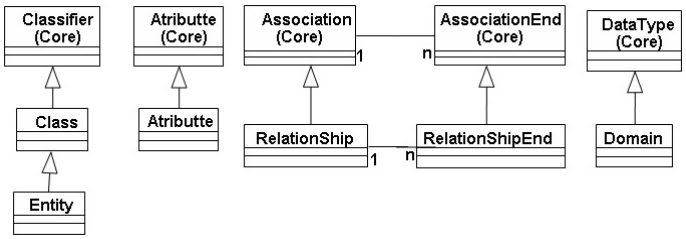


Figura 5.2. Relación entre las clases del metamodelo UML y el metamodelo ER

El metamodelo ER (figura 5.3) contiene todos los elementos de modelado de ER: entidad, relación, etc. En el metamodelo ER, una entidad (*Entity*) se compone por un conjunto de atributos (*Attribute*). Una relación (*RelationShip*) representa una asociación entre dos o más entidades. Cada rol (*RelationShipEnd*) de una entidad en una relación, puede tener de manera opcional nombre y multiplicidad.

A pesar de la importancia del metamodelo ER no existe una notación estándar que le dé soporte. En esta tesis,

emplearemos la notación más cercana a la propuesta por Chen [CPP76].

En la figura 5.4 se resume el conjunto de símbolos que utilizaremos y la relación que guardan con los conceptos definidos en el metamodelo ER.

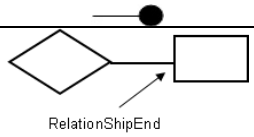
Concepto	***Icono***
Entity	
RelationShip	
Attribute	
RelationShipEnd	

Figura 5.4. Relación entre conceptos e iconos del metamodelo ER

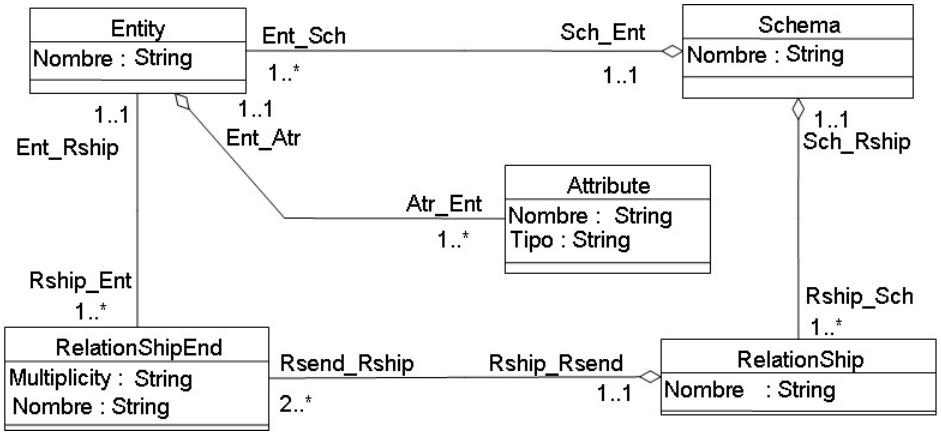


Figura 5.3. Metamodelo Entidad Relacional de CWM

5.2.2. Metamodelo OLAP.

El metamodelo OLAP (figura 5.5) contiene todos los conceptos y elementos de modelado multidimensional presentados en el capítulo 2 [OMG03a].

En el metamodelo OLAP, un esquema (*Schema*) contiene dimensiones (*Dimensions*) y cubos (*Cubes*). Una dimensión se compone de niveles (*Levels*) y jerarquías (*Hierarchies*).

Un cubo, es una colección de medidas (*Measures*). La clase *CubeDimensionAssociation* relaciona un cubo con una dimensión. La clase *HierarchyLevelAssociation*, relaciona un nivel con una jerarquía

Para representar gráficamente las instancias del metamodelo OLAP (modelos multidimensionales) emplearemos diagramas de clases en UML, donde a cada elemento del diagrama le asociaremos un estereotipo textual que relacione el elemento con el concepto del metamodelo que representa en la instancia.

En la figura 5.6 se muestra el conjunto de estereotipos definidos y su relación con los conceptos del metamodelo OLAP. En un modelo multidimensional, el concepto de jerarquía no se representa explícitamente, se encuentra implícito en la representación de las asociaciones entre los niveles que componen la jerarquía, por lo que no se utiliza ninguna representación explícita para este concepto del metamodelo.

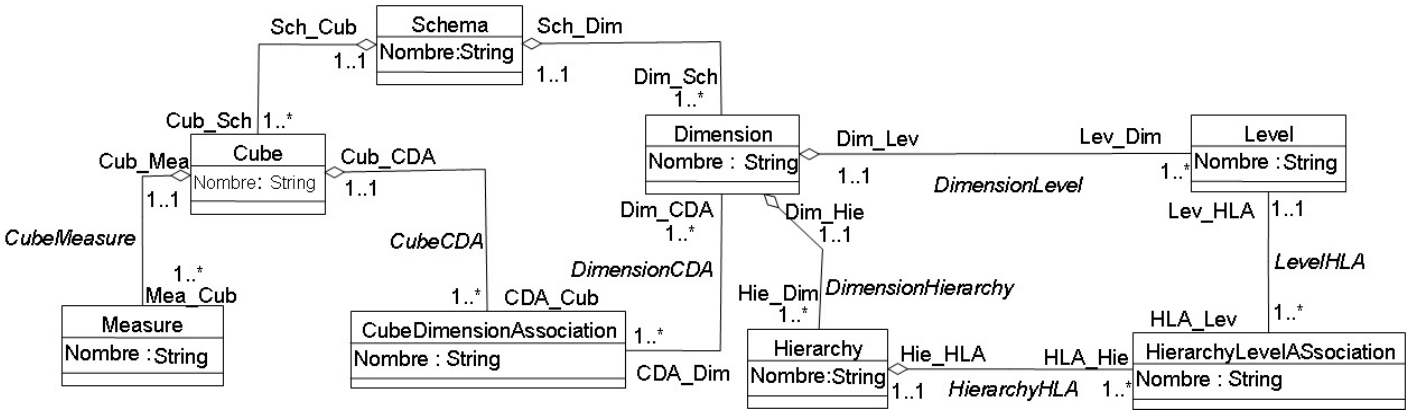


Figura 5.5. Metamodelo OLAP de CWM

Metamodelo OLAP	*Representación UML*
Clase Cube	*Clase* <<Cube>>
Clase Dimension	*Clase* <<Dimension>>
Clase CubeDimensionAssociation	*Asociación* <<CDA>>
Asociación CubeCDA	
Asociación DimensionCDA	
Clase Level	*Clase* <<Level>>
Asociación DimensionLevel	....
Clase Hierarchy	...
Asociación DimensionHierarchy	*Asociación <<DH>>*
Clase HierarchyLevelAssociattion	*Asociación* <<HLA>>
Asociación HierarchyHLA	
Asociación LevelHLA	
Clase Measure	*Atributo* <<M>>
Asociación CubeMeasure	

Figura 5.6. Relación entre los conceptos del metamodelo OLAP y los estereotipos UML definidos

En el metamodelo OLAP, la asociación *DimensionLevel* que relaciona una dimensión con sus niveles, se puede derivar de la asociación *DimensionHierarchy* y de la clase *HierarchyLevelAssociattion*, y sus asociaciones *HierarchyHLA* y *LevelHLA*, que relacionan respectivamente una jerarquía con la dimensión a la que pertenece y con los niveles que la forman, por lo que esta asociación *DimensionLevel* no será representada explícitamente en nuestra representación UML de los modelos OLAP.

Las jerarquías de niveles se representan en el metamodelo OLAP por la clase *HierarchyLevelAssociattion* y sus asociaciones *HierarchyHLA* (ordenada) y *LevelHLA*, pueden representarse de forma equivalente por un tipo de asociación (*HLA*) definido reflexivamente sobre la clase *<<Level>>*, que define la relación de orden entre niveles que componen una jerarquía, por ello en nuestra representación UML de los modelos OLAP, estos tres tipos *HierarchyHLA*, *HierarchyLevelAssociattion* y *LevelHLA* no se representan

directamente, siendo sustituidos por el tipo de asociación <<*HLA*>>

5.3. Fase 1: derivación de modelos multidimensionales.

El objetivo de esta fase, consiste en identificar los modelos candidatos multidimensionales implícitos en el modelo conceptual de la base de datos operacional. Para identificar el conjunto de modelos multidimensionales, es necesario que el modelo ER sea reestructurado.

La reestructuración, consiste en eliminar las relaciones de herencia y transformar las relaciones que por su semántica son candidatas a ser cubos de datos en el modelo OLAP.

El proceso de obtención del conjunto de modelos multidimensionales a partir del modelo ER (reestructurado) de la base de datos operacional, consiste en realizar un análisis de dicho modelo, con el fin de identificar las entidades que son candidatas a ser cubo de datos.

Una vez identificadas éstas, se aplica un conjunto de reglas de transformación horizontales sobre la instancia (modelo) del metamodelo ER (metamodelo fuente) para producir un conjunto de instancias (modelos) del metamodelo OLAP (metamodelo destino). El proceso que proponemos para obtener el conjunto de modelos multidimensionales candidatos, se compone de dos pasos:

- Reestructuración del modelo ER.
- Transformación del modelo ER en un modelo OLAP.

5.3.1. Reestructuración del modelo ER.

Este paso, consiste en reorganizar el modelo ER con el objeto de tener un modelo con una estructura uniforme y poder aplicar el conjunto de reglas de transformación de una manera más sencilla. Al finalizar este paso, se obtiene un nuevo modelo ER en el que las relaciones de herencia y las relaciones binarias muchos a muchos clasificadas como cubos de datos candidatos se han reestructurado.

La reestructuración consiste en dos pasos:

a) Identificación y transformación de cubos de datos candidatos a partir de relaciones binarias muchos a muchos.

b) Transformación de jerarquías de herencia.

a) Identificación y transformación de cubos de datos candidatos a partir de relaciones binarias muchos a muchos.

Este paso, tiene por objetivo identificar en el modelo ER los cubos de datos candidatos que pueden derivarse de relaciones binarias con multiplicidad muchos a muchos. Generalmente, un cubo de datos del modelo OLAP procede de una entidad del modelo ER, sin embargo, existen ocasiones donde un cubo de datos se obtiene de relaciones binarias de multiplicidad muchos a muchos.

La elección de una relación binaria como cubo de datos candidato no depende de ninguna propiedad estructural, depende de un aspecto semántico. Por ello, no es posible definir una condición a nivel estructural que permita

identificar cuándo una relación binaria representa un cubo de datos. La clasificación de una relación como cubo candidato debe ser guiada por el significado que tiene la relación en el modelo. Este proceso, debe realizarse de manera manual y está sujeto al conocimiento que el analista tenga sobre el modelo ER.

Las relaciones binarias con multiplicidad muchos a muchos que hayan sido identificadas como cubos de datos candidatos, deben transformarse en dos relaciones de multiplicidad muchos a uno entre las entidades originales y una nueva entidad que permita conservar la asociación original. Por ejemplo, al lado izquierdo de la figura 5.7 se muestra el modelo ER correspondiente a una base de datos operacional, donde aparece una relación binaria con multiplicidad muchos a muchos.

Al lado derecho, se muestra el modelo reestructurado. En el modelo reestructurado, la nueva entidad será considerada entidad de cubo candidata.

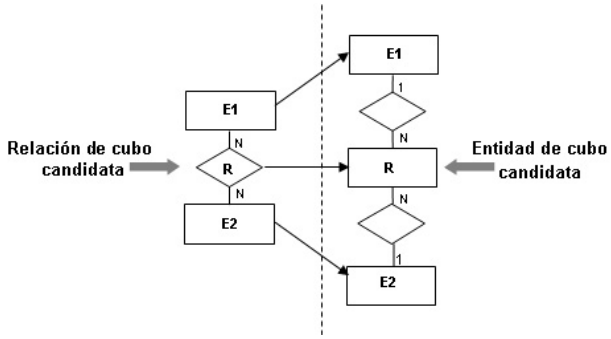


Figura 5.7. Relaciones binarias muchos a muchos

b) Transformación de jerarquías de herencia

Si el modelo conceptual de la base de datos operacional contiene relaciones de herencia, dichas relaciones serán colapsadas.

La entidad que representará la relación de herencia coincide con la entidad raíz, los atributos de las entidades hijas se convierten en atributos de dicha entidad, añadiendo un atributo discriminante que indique el tipo de especialización de cada ocurrencia.

Por ejemplo, al lado derecho de la figura 5.8, se muestra la entidad generada durante el proceso de reestructuración.

El nombre de esta entidad coincide con el nombre de la entidad raíz de la jerarquía en el modelo original. Los atributos de la nueva entidad, se obtienen a partir de los atributos de las entidades que participan en la relación de herencia más el atributo discriminante (D).

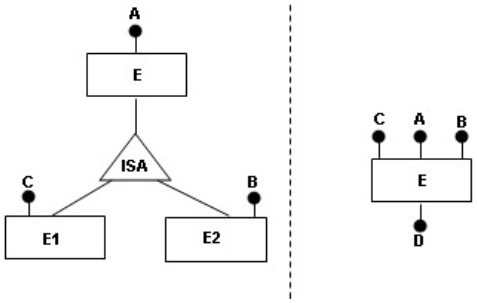


Figura 5.8. Relaciones de herencia

5.3.2. Transformación del modelo ER en un modelo OLAP.

Un proceso de transformación consiste en derivar un modelo destino *S'* a partir de un modelo fuente *S,* sustituyendo cada elemento *C* de *S* por un nuevo elemento *C'* de *S'*.

Formalmente, una transformación *Σ* se define como una tupla <*T , t*> tal que: *C' = T(C)* y *c' = t(c)*, donde c es una instancia de *C* y *c'* una instancia de *C'*. Una correspondencia estructural *T* define la relación semántica entre dos elementos de los modelos que participan en la transformación.

Nosotros identificamos dos tipos de correspondencias estructurales para obtener un modelo OLAP a partir de un modelo ER: simples y complejas.

Correspondencias simples: estas correspondencias definen la relación semántica entre un elemento del modelo ER y un elemento del modelo OLAP.

Correspondencias complejas: estas correspondencias definen una relación semántica entre dos o más elementos del modelo ER y un único elemento en el modelo OLAP.

A continuación se describe el conjunto de correspondencias:

1. Obtención de cubos de datos.

Definición: *Una entidad de cubo candidata es una entidad del modelo ER que hace referencia*[6] *a otras entidades pero que no es referida por ninguna otra.*

- *Sea* H una entidad de cubo candidata con un atributo numérico A (figura 5.9), entonces:

  1.1. H *se transforma en un cubo de datos del mismo nombre en el modelo OLAP.*

  1.2. A *se transforma en una medida del mismo nombre del cubo de datos* H *en el modelo OLAP.*

2. Obtención de dimensiones.

- Sea *R* una relación binaria entre las entidades *H* y *E* del modelo ER, tal que *H* es una entidad de cubo candidata, la multiplicidad de la relación es *R(H(1,1), E(0,N))*[7] y sean *h,* y *e* respectivamente los roles de *H* y *E* en la relación *R* (figura 5.10), entonces:

---

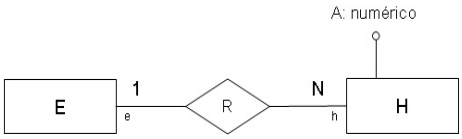
[6] Se dice que una entidad E_i hace referencia a otra entidad E_j si existe una relación binaria R_{ij} entre ambas de multiplicidad R_{ij} (E_i (1,1), E_j (0,N)).

[7] No se admiten esquemas multidimensionales con relaciones M:M entre el hecho y alguna de sus dimensiones.

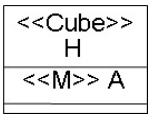
2.1. R *se transforma en una clase <<Dimensión>> del mismo nombre, en el modelo OLAP.*

2.2. h *se transforma en una asociación <<CDA>> del mismo nombre, entre el cubo de datos* H *y la dimensión* R *en el modelo OLAP.*

Modelo ER



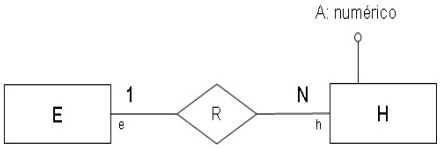
Modelo OLAP



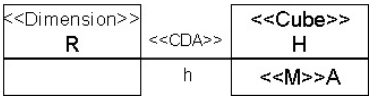
La entidad *H* del modelo ER, se transforma en un cubo de datos, el atributo *A* se transforma en una medida del cubo *H*.

Figura 5.9. Transformación de una entidad en cubo

Modelo ER



Modelo OLAP



La relación *R* del modelo ER, se transforma en una clase <<Dimensión>> del mismo nombre, el rol *h* se transforma en una asociación <<CDA>> del mismo nombre entre el cubo *H* y la dimensión *R*.

Figura 5.10. Transformación de una relación en dimensión

3. Obtención de niveles y jerarquías.

Definición**:** Un camino *c* en un modelo ER, es una secuencia de entidades $c=(E_1, E_2,\ldots, E_n)$ tal que existe una secuencia de relaciones binarias en el modelo, $R_1(H, E_1)$, $R_2(E_1, E_2)$,..., $R_n(E_{n-1}, E_n)$, donde *H* es una entidad de cubo candidata y no existe ninguna relación *R* de la forma $R(E_n,-)$.

- *Sea* c *un camino del modelo ER (figura 5.11):*

  3.1. Toda entidad E_i de c se transforman en una clase <<Level>> del mismo nombre (de la dimensión R_1), en el modelo OLAP.
  3.2. c se transforma en una jerarquía del mismo nombre, (de la dimensión R_1), en el modelo OLAP
  3.3. El rol de la entidad E_1 en la relación $R_1(H, E_1)$ se transforma en una asociación <<DH>> entre la jerarquía c y la dimensión R_1.
  3.4. Toda relación R_i (i:2..n) de c se transforma en una asociación <<HLA>> entre los niveles E_{i-1} y E_i, en el modelo OLAP.

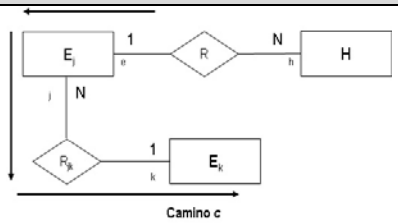
Definición de las reglas de transformación en QVT.

En esta sección se describen las reglas de transformación empleando la notación gráfica del lenguaje relacional de QVT[8] [OMG05b]. Esta notación permite realizar una representación visual de cada transformación.

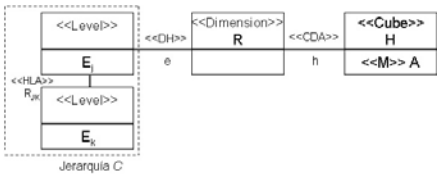
En el lenguaje relacional de QVT, una transformación entre modelos candidatos se especifica por un conjunto de relaciones que deben ejecutarse para que la transformación se realice con éxito. Un modelo candidato es cualquier modelo de un tipo de metamodelo.

[8]Debido a que este lenguaje se encuentra actualmente, en desarrollo [AVK06]. En algunas reglas, realizamos adecuaciones sintácticas para poder expresar la semántica requerida.

Modelo ER



Modelo OLAP



El camino *c* formado por las entidades E_j y E_k se transforma en la jerarquía *c* de la dimensión *R*. La relación R_{jk}, se transforma en una asociación <<HLA>> entre los niveles E_j y E_k de la jerarquía *c.* El rol *e* se transforma en una asociación <<DH>> entre la jerarquía *c* y la dimensión *R*, en el modelo OLAP.

Figura 5.11. Transformación niveles y jerarquías

Al igual que en los lenguajes de programación, un tipo de datos define los valores válidos para una variable en el lenguaje, un modelo candidato se ajusta a los valores permitidos por el metamodelo.

Las relaciones en la transformación, definen restricciones que deben cumplir los elementos de los modelos. Una relación, está compuesta por dos o más dominios y un par de cláusulas opcionales *When* y *Where*.

En la cláusula *Where* se define la condición que debe ser satisfecha por los elementos de los modelos que participan en la relación, mientras que en la cláusula *When* se establece la condición que debe cumplirse antes de la transformación. Estas cláusulas pueden estar acompañadas por un conjunto de instrucciones en el lenguaje OCL.

Un dominio consta de un conjunto de elementos de un metamodelo. Un dominio presenta un patrón, que puede verse como un grafo de elementos del metamodelo con sus propiedades, asociaciones y nombres. En una relación, se pueden definir dos tipos de dominios: dominio de comprobación (*Checkonly(C)*) y dominio obligado (*Enforcable (E)*).

Un dominio de comprobación sirve para comprobar que el modelo candidato satisface el patrón. En cambio, un dominio obligado asegura que en la transformación los elementos generados serán creados en el modelo destino de acuerdo al patrón. Además, para cada dominio se indica el nombre de su metamodelo.

En la figura 5.12 se muestra la notación gráfica para representar una relación. La notación *er: ENTREL*[9] y *md: OLAP* en los extremos de la relación indica una transformación entre dos modelos candidatos *er* y *md,* asociados respectivamente a los metamodelos *ENTREL* y *OLAP*. El tipo de dominio se denota con las letras *C* (dominio de comprobación) o *E* (dominio obligatorio).

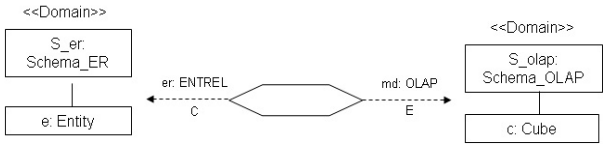


Figura 5.12. Representación gráfica de una relación

Regla *Main* (principal).

El proceso de transformación, se inicia con la regla de transformación *Main* (figura 5.13*)*.

En la representación gráfica de esta regla se muestran los modelos candidatos *er* y *md*, asociados a los metamodelos *ENTREL* y *OLAP* respectivamente.

El objetivo de esta regla, es transformar un modelo ER en un modelo OLAP. Esta regla, se compone de las reglas de transformación *EntityToCube()* (cláusula *Where*) y *RelationShipToDimension()*. Estas reglas, se encargan de transformar las entidades de cubo candidatas (e) y las relaciones (r) del modelo ER, en cubos (c) y dimensiones (d) del modelo OLAP respectivamente.

[9] Entidad-Relación

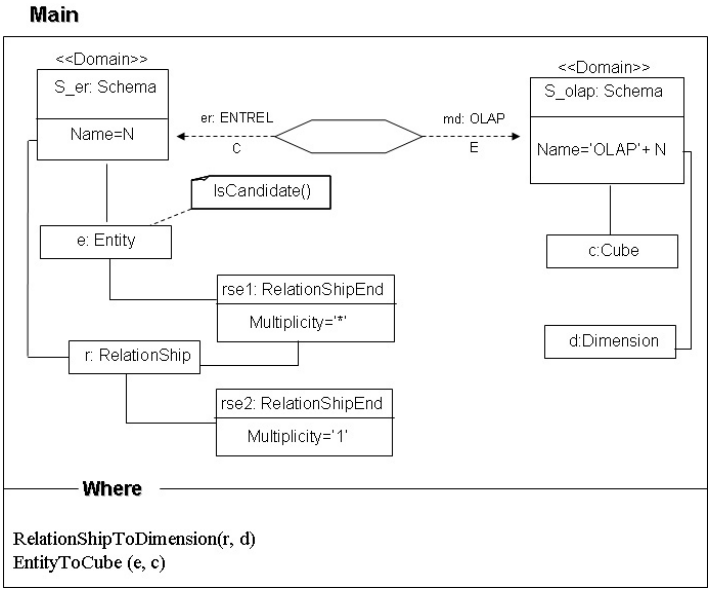


Figura 5.13. Regla *Main* (principal)

El dominio de comprobación establece las siguientes restricciones sobre los elementos del modelo ER que participan en la transformación:

- Las entidades a ser transformadas en un cubo de datos, deben ser entidades de cubo candidatas. Para identificar las entidades de cubo candidatas, se utiliza la función OCL *IsCandidate()*.
- Las relaciones a ser transformadas en dimensiones, deben tener multiplicidad muchos a uno, además el rol con multiplicidad igual a muchos debe estar asociado a la entidad de cubo candidata.

Regla EntityToCube.

El objetivo de esta regla, es convertir una entidad de cubo candidata (e) del modelo ER en un cubo de datos (c) del modelo OLAP. En la representación gráfica de la regla (figura 5.14), se muestra que un cubo (c), se crea por medio de un dominio obligado. Esta regla de transformación, se compone de las reglas *AttributeToMeasure()* y *RelationShipEndToCDA()* (cláusula *Where*). Estas reglas se encargan de trasformar los atributos (a) y los roles (rse) de la entidad (e) en medidas (m) y clases de asociación (cda) del modelo OLAP respectivamente.

El dominio de comprobación, establece las siguientes restricciones sobre los elementos que participan en la transformación:

- los atributos, deben ser de tipo numérico.
- los roles, deben tener multiplicidad igual a muchos.

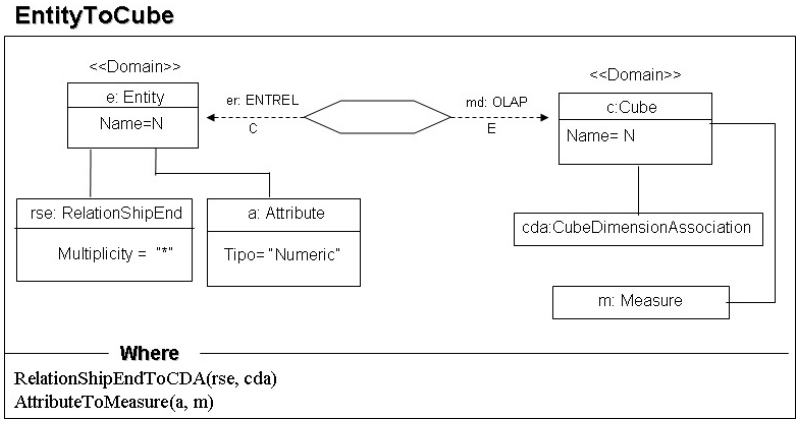


Figura 5.14. Regla *EntitytoCube*

Regla AttributeToMeasure.

El objetivo de esta regla, es transformar un atributo de una entidad de cubo candidata, en el modelo ER, en una medida de cubo den el modelo OLAP.

En la representación gráfica de la regla (figura 5.15), se muestra que una medida (m), se crea por medio de un dominio obligado. El nombre de la medida (m), se crea a partir del nombre del atributo (a).

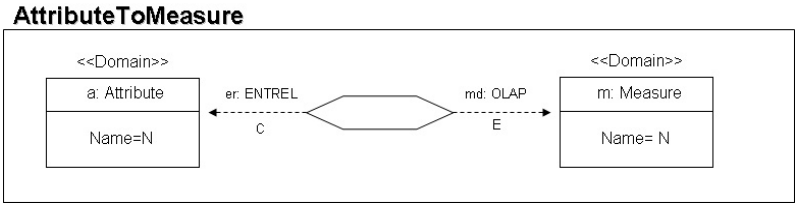


Figura 5.15. Regla *AttributeToMeasure*

Regla RelationShipToDimension.

El objetivo de esta regla, es convertir una relación (r) del modelo ER en una dimensión (d) del modelo OLAP. En la representación gráfica de esta regla (figura 5.16), se muestra que la dimensión se crea por medio de un dominio obligado y que el nombre de la dimensión se forma a partir del nombre de la relación. Esta regla, se compone de las reglas de transformación *RelationShipEndToDimHierarchy*, *EntityToLevel* y *HierauxToHierarchy* y por las funciones de OCL *Genera_jerarquías()* y *Genera_Niveles(),* que se describen a continuación:

- La regla *RelationShipEndToDimHierarchy*, permite transformar el rol (rse1) de una relación en una asociación dimensión-jerarquía (*DimensionHierarchy*).

- La función *Genera_Jerarquías()*, permite identificar los distintos caminos (J), que se forman a partir de la relación transformada en dimensión. El resultado de esta función se almacena en la variable *HierarchyAux* (cláusula *Where* de la figura 5.16).

  En el ejemplo de la Figura 5.17, se muestra que a través de las relaciones muchos a uno de la relación *r1,* se identifican dos caminos (camino 1 y camino 2).

- La regla *HierAuxToHierarchy()*, tiene por objetivo trasformar cada elemento almacenado en la variable *HierarchyAux* (es decir cada camino) en una jerarquía (h) del modelo OLAP. Continuando con nuestro ejemplo, los caminos 1 y 2 (figura 5.17), serán transformados en jerarquías de dimensión por ésta regla de transformación.

- La función *Genera_Niveles()*, permite identificar las entidades del esquema ER que serán transformadas en niveles (l) de dimensión.

  Estas entidades se obtienen de los caminos almacenados en la variable *HierarchyAux* (cláusula *Where* de la figura 5.16). En el ejemplo de la Figura 5.17 se identifican las entidades *e1, e2 y e3* en los caminos 1 y 2.

  El resultado de esta función se almacena en la variable *LevelAux* (cláusula *Where* de la figura 5.16).

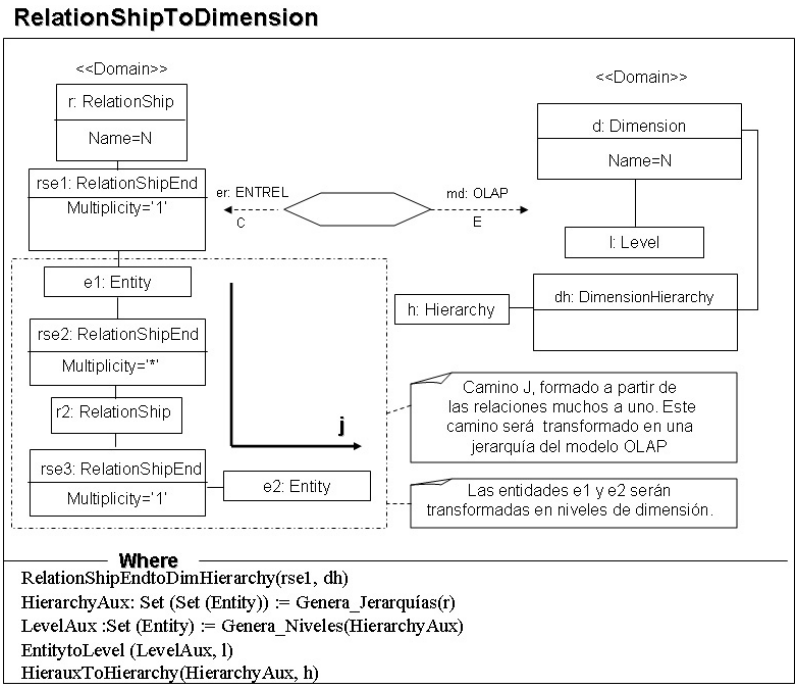


Figura 5.16. Regla RelationShipToDimension

- La regla de transformación *EntityToLevel()*, tiene por objetivo transformar cada entidad almacenada en la variable *LevelAux* en un nivel de dimensión. Las entidades *e1, e2 y e3* de nuestro ejemplo, serán transformadas en niveles de dimensión.

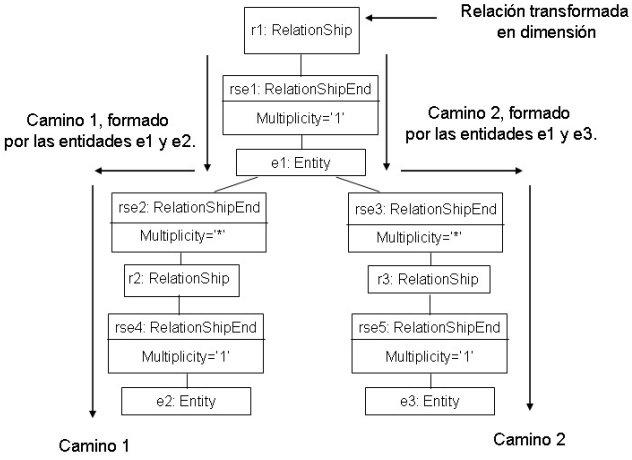


Figura 5.17. Jerarquías y niveles en el modelo ER

Regla HierAuxToHierarchy.

El objetivo de esta regla es transformar una secuencia de entidades en el modelo ER en una jerarquía (h) de dimensión en el modelo OLAP.

Esta regla se compone de la regla *RelationShipToHLA()* y de la función OCL *Genera_nombre_jerarquía()*.

El objetivo de la función OCL es generar un nombre para la secuencia que será utilizado para formar el nombre de la jerarquía (figura 5.18).

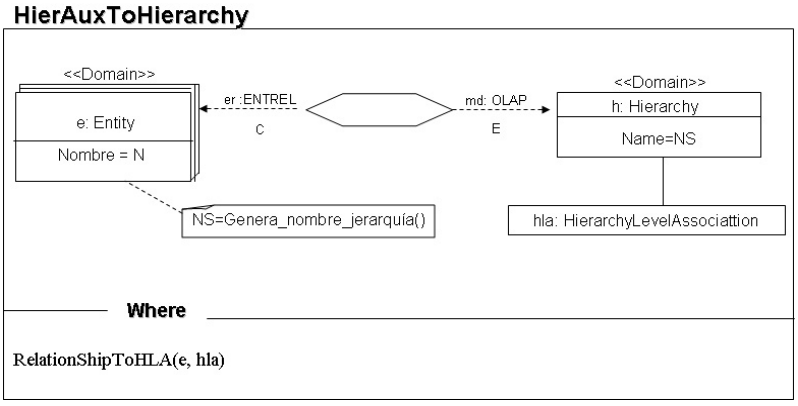


Figura 5.18. Regla *HierAuxToHierarchy*

Regla RelationShipToHLA.

El objetivo de esta regla es asociar los niveles de dimensión con sus jerarquías. En la representación gráfica de la regla (figura 5.19), se muestra que una clase de asociación, (*hla)*, se crea por medio de un dominio obligado.

El nombre de la asociación coincide por el nombre de la relación.

La regla *RelationShipToHLA(),* se ejecutará sólo cuando se cumpla la precondición *EntityToLevel()* (cláusula *When*) con lo cual, aseguramos la correcta asociación de un nivel con la jerarquía.

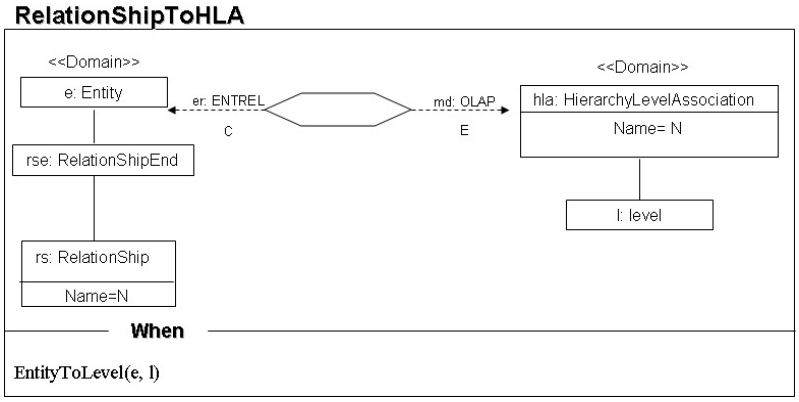


Figura 5.19. Regla *RelationShipToHLA*

Regla EntityToLevel.

El objetivo de esta regla es transformar una entidad (e) en un nivel (l) de dimensión del modelo OLAP.

En la representación gráfica de esta regla (figura 5.20) se muestra que por medio de un dominio obligado se crea el nivel de dimensión y que el nombre del nivel se forma con el nombre de la entidad.

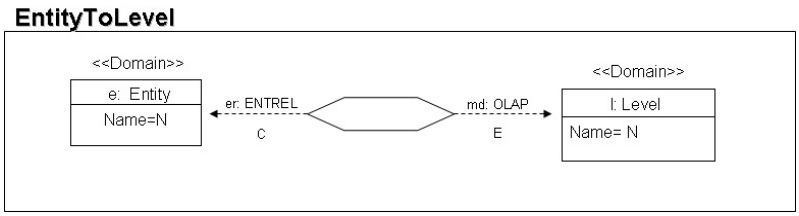


Figura 5.20. Regla *EntityToLevel*

Regla RelationShipEndToCDA.

El objetivo de esta regla es convertir un rol (*RelationShipEnd*) asociado a una entidad de cubo candidata en una clase de asociación entre cubo y dimensión (*CubeDimensionAssociation*) del modelo OLAP.

En la representación gráfica de la regla (figura 5.21) se muestra como la clase *CubeDimensionAssociation,* se crea por medio de un dominio obligado.

El nombre de la clase de asociación se compone por el nombre de la clase *RelationShipEnd*.

La regla *RelationShipEndToCDA()* se ejecutará sólo cuando se cumpla la precondición *RelationshipToDim()* (cláusula *When*) con lo cual aseguramos la correcta asociación de un cubo con sus dimensiones.

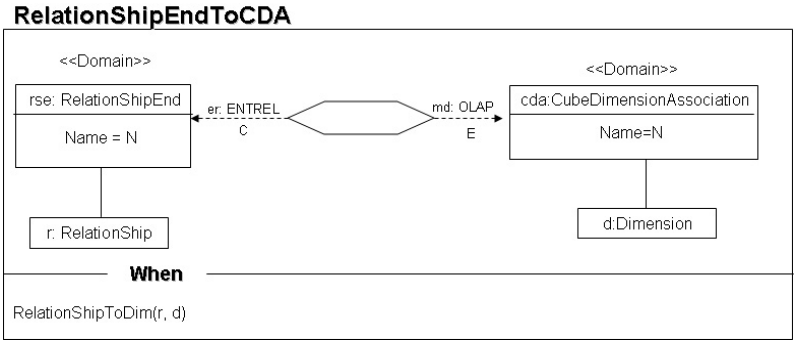


Figura 5.21. Regla *RelationShipEndToCDA*

Regla RelationShipEndToDimHierarchy.

El objetivo de esta regla es convertir el rol (rse) en una asociación dimensión-jerarquía (dh) del modelo OLAP.

En la representación gráfica de la regla (figura 5.22) se muestra que una asociación dimensión-jerarquía (dh) se crea por medio de un dominio obligado. El nombre de la asociación se crea a partir del nombre del rol (rse).

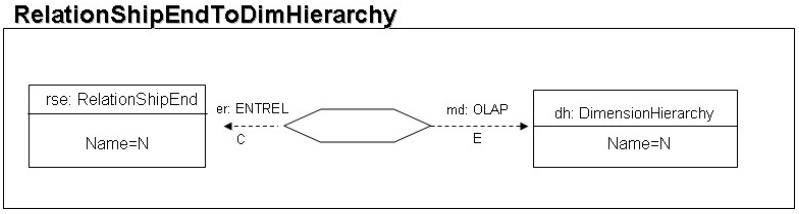


Figura 5.22. Regla RelationShipEndToDimHierarchy

5.3.3. Caso de Estudio.

En este apartado, se muestra el proceso de transformación de un modelo ER en un modelo OLAP , para ello emplearemos el modelo de la figura 5.23. El primer paso consiste en identificar los cubos de datos candidatos que se pueden derivan a partir de relaciones binarias con multiplicidad muchos a muchos. En el esquema de la figura 5.23, el cubo de datos candidato que se identificada es: *Línea*. Una vez identificados, es necesario reestructurar el esquema ER, con la finalidad de eliminar jerarquías de herencia y las relaciones clasificadas como cubos de datos candidatos.

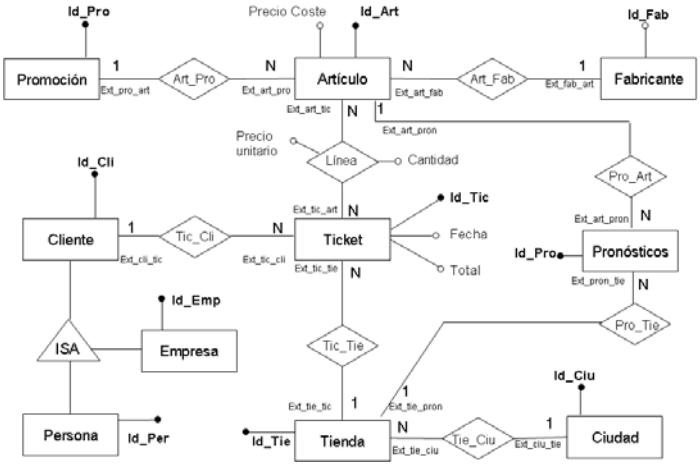
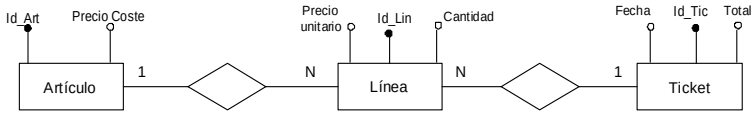
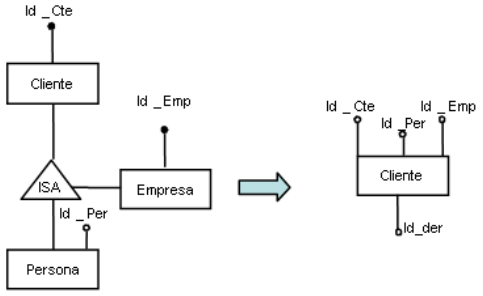


Figura 5.23 Modelo ER

En la figura 5.24, se muestran los sub-esquemas reestructurados para la relación *Línea* y la jerarquía de herencia.



a). Sub-esquema Línea reestructurado



b) Relación de herencia reestructurada

Figura 5.24 Reestructuración del esquema ER

Una vez reestructurado el esquema ER, es necesario identificar las entidades de cubo candidatas. De acuerdo a la condición establecida en la sección 5.3.3 en el esquema ER de la figura 5.25, se identifican dos entidades de cubo candidatas: *Línea* y *Pronósticos*.

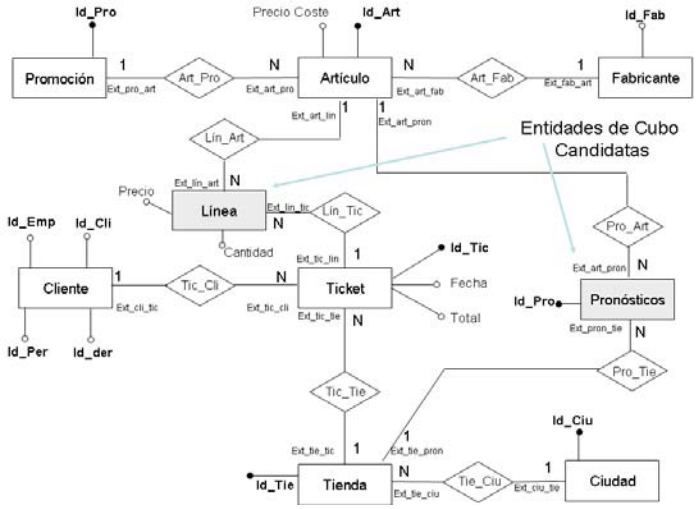


Figura 5.25 Esquema ER reestructurado

El proceso de transformación, se inicia a partir de cada entidad de cubo candidata.

En la figura 5.26, se muestra una transformación parcial para la entidad de cubo candidata *Línea*. El resultado de la primera transformación, lo define la regla de transformación *EntityToCube*, donde la entidad de cubo candidata *Línea* se transforma en el cubo de datos del mismo nombre. De acuerdo a esta regla de transformación, las reglas *AttributeToMeasure* y *RelationShipEndToCDA* también deben realizarse.

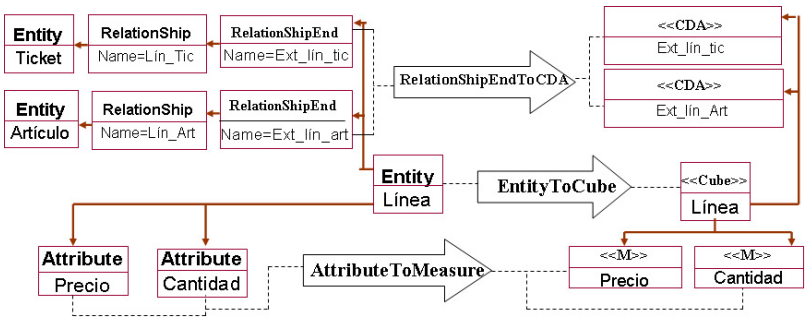


Figura 5.26 Proceso de transformación a partir de *Línea*

El proceso de transformación para las reglas de transformación *EntityToLevel* y *RelationShipToDimension* se muestran en la figura 5.27.

La dimensión del cubo de datos *Línea*, se genera a partir de la relación *Lín_Tic*. Mientras que las entidades *Ticket*, *Cliente* y *Tienda* se transforman en niveles de la dimensión *Lin*_Tic. El rol *Ext_tic_lin*, se transformó en una asociación dimensión-jerarquía (<<DH>>). Las relaciones *Tic_CLi* y *Tic_Tie*, se convierten en clase de asociación *HierarchyLevelAssociation* (figura 5.28).

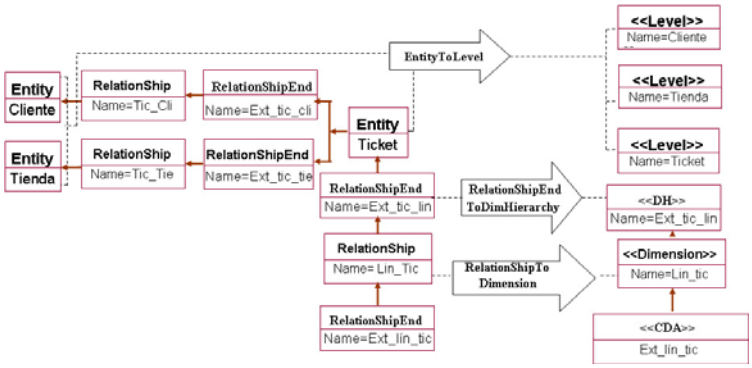


Figura 5.27 Generación de dimensiones y niveles

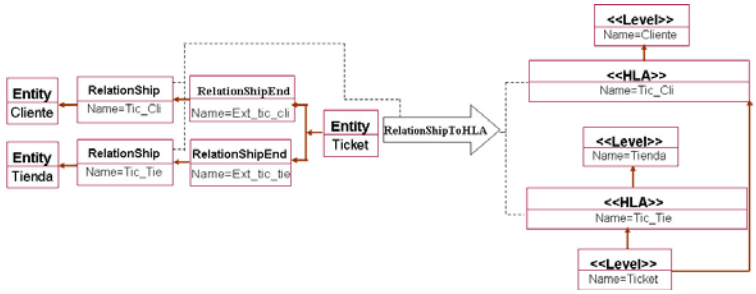
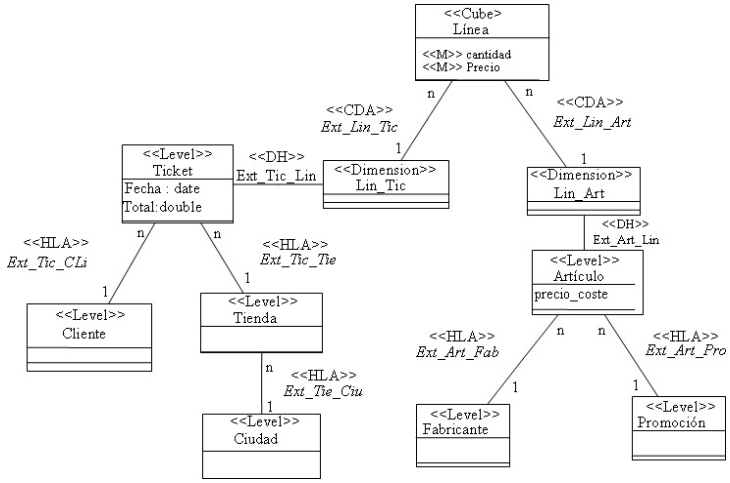
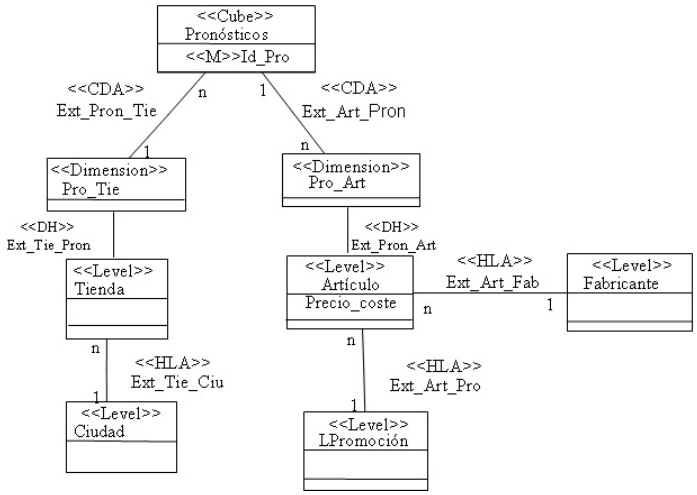


Figura 5.28. Generación de niveles

Los esquemas multidimensionales candidatos generados por el proceso de transformación, se muestran en la figura 5.29.



b) Esquema candidato Línea



a) Esquema candidato Pronóstico

Figura 5.29. Esquemas multidimensionales candidatos

5.4. Fase 2: Especificación de requisitos de usuario.

En esta sección, se muestra el modelo de requisitos para capturar los requisitos de un AD. Este modelo se fundamenta en la filosofía de las metodologías basadas en metas, la cual ha sido ampliamente aceptada para la captura de requisitos en sistemas tradicionales.

El modelo de requisitos presentado en este capítulo, permite capturar los conceptos del modelo multidimensional a partir de las tareas que el usuario pretende realizar al interactuar con el AD.

Posteriormente, se describe el conjunto de tareas por medio de casos de uso y se representa la interacción del usuario con el sistema por medio de diagramas de actividad. Por

último, a partir de los diagramas de actividad, se obtienen la información que el AD deberá almacenar.

La información generada en esta fase permitirá la selección y el refinamiento de los esquemas multidimensionales candidatos que mejor satisfagan los requisitos de usuario.

5.4.1. Descripción del modelo de requisitos.

El modelo de requisitos propuesto, cubre el análisis de los requisitos y la especificación de los requisitos de información.

El análisis de los requisitos inicia con una lista de los usuarios y las tareas de análisis que pretende realizar al interactuar con el AD. Las tareas son integradas en metas para formar un Árbol de Refinamiento de Metas (ARM).

A partir de las metas representadas en el ARM, se identifican los conceptos relacionados con el dominio del problema que permitirán la selección y el refinamiento de los esquemas multidimensionales candidatos.

Inspirados en el trabajo propuesto por [Ins03], el esquema del diseño propuesto es dividido en tres etapas: *definición de la misión, identificación de las metas de análisis y especificación de los requisitos*.

Los usuarios del sistema de AD, plantean el objetivo principal que debe ser satisfecho (misión). Las interacciones entre los usuarios y el sistema, deben de dar respuesta a este propósito.

El ARM representa las tareas de análisis y facilita la comunicación del conocimiento del problema entre los

desarrolladores y los usuarios finales. Los requisitos por otra parte, representan los datos que deben formar parte del AD.

Etapa 1: Definición de la misión

La primera etapa dentro del modelo de requisitos, consiste en definir la misión del AD. La misión, se refiere a una descripción breve sobre el propósito del sistema que se pretende construir, así como de los servicios que éste proporcionará a sus futuros usuarios **[**Ins03**].**

Las razones para enunciar la misión del AD, son dos. En primer lugar, los usuarios, diseñadores, programadores y todos los involucrados en el desarrollo del sistema tendrán en mente el objetivo último del sistema. En segundo lugar, fijar el objetivo último del sistema evitará diferencias de interpretación sobre el propósito del mismo. El enunciado de la misión, debe estar compuesto al menos por dos elementos:

***Definición*.-** Consiste en una breve descripción del sistema que se pretende desarrollar.

***Propósito*.-** Consiste en la definición del objetivo del sistema.

Las técnicas que pueden utilizarse durante esta etapa son: entrevistas, reuniones en grupo, cuestionarios o inmersión en el negocio del cliente, entre otras. Para ilustrar esta etapa, continuaremos con el caso de estudio introducido en la sección 5.3.4.

Caso de estudio (Continuación):

***Misión*:** el sistema proporcionará los medios necesarios para analizar la organización**.**

***Propósito*:** el objetivo del sistema, será proporcionar información relativa a la conducta de los clientes, el rendimiento de la empresa y el funcionamiento de cada sucursal.

Etapa 2: Identificación de las metas de análisis

El objetivo principal de esta etapa, consiste en identificar las tareas y metas que el usuario llevará a cabo con el AD. En un ambiente de AD estas tareas, se relacionan con los procesos de negocios a ser analizados.

Siguiendo los lineamientos de las metodologías orientadas a metas, nosotros partimos de un conjunto de tareas específicas desempeñadas por los usuarios del AD y posteriormente realizamos la integración de las tareas hasta llegar a tareas más generales (metas).

Para lograr esto nosotros proponemos dos pasos: 1) definición de los procesos de negocio. 2) identificación de las metas de análisis.

Paso 1. Definición de los procesos de negocio.

El primer paso, tiene por objetivo definir los procesos de negocio que van a ser analizados [KR02].

El análisis de los procesos de negocio, proporcionará al usuario el conocimiento necesario para la toma de decisiones estratégicas que permitan mejorar dicho proceso de negocio.

Para identificar los procesos de negocio, se pueden utilizar técnicas tradicionales de obtención de requisitos como son los grupos de trabajos o las entrevistas [BNRH88].

Caso de estudio (Continuación): en nuestro ejemplo, supondremos que el proceso de negocio a analizar son las ventas (*Ventas)***.**

Con la finalidad de realizar una descripción más precisa del proceso de negocio y lograr la especificación de los requisitos del AD, es necesario indicar las tareas de análisis específicas que el usuario realizará al interactuar con el AD.

Por ejemplo, algunas de las tareas específicas relacionadas con el proceso de negocio *Ventas* pueden ser: analizar los canales de entrega de mercancías o bien, analizar el impacto de la presentación en las ventas.

En la siguiente sección, se explica una estrategia para identificar las tareas de análisis a realizar sobre el proceso de negocio.

Paso 2. Identificar las metas de análisis

La finalidad de este paso, consiste en describir el funcionamiento del AD en relación a las tareas de análisis que los usuarios pretenden realizar al interactuar con él.

Nuestra propuesta, propone la construcción de un Árbol de Refinamiento de Metas (ARM). Para lo cual, usamos los conceptos de meta y tarea empleados por las metodologías orientadas a metas [Lam01].

La construcción del ARM, puede ser realizada empleado técnicas de refinamiento o de abstracción [Ant96] [EPM04]. A continuación se explica brevemente cada una de ellas:

Refinamiento de metas: utiliza un enfoque *top-dow* para la identificación de las metas. Se parte de las metas de más

alto nivel de la organización hasta llegar a las tareas individuales realizadas por los actores del proceso de negocio.

Una vez que una meta organizacional es detectada, es necesario refinarla hasta alcanzar objetivos que puedan ser ejecutados por actores organizacionales individuales es decir, cuando se alcance el nivel de tareas que puedan ser satisfechas mediante tareas únicas de alguno de los actores organizacionales.

Abstracción de metas: utiliza un enfoque bottom-up para identificar las metas. Se inicia con las tareas de bajo nivel de los actores organizacionales y se procede a encontrar las metas del negocio con las que éstas se corresponden.

Para identificar las metas, es necesario precisar primero quienes son los usuarios responsables de llevar a cabo las tareas.

Sin importar la estrategia seguida para la construcción del ARM, la estructura de éste se compone, por un nodo raíz que representa la meta de más alto nivel, nodos intermedios que representan metas y por último las tareas.

En la figura 5.30, se muestra un ARM y la notación gráfica que empleamos para la construcción del ARM.

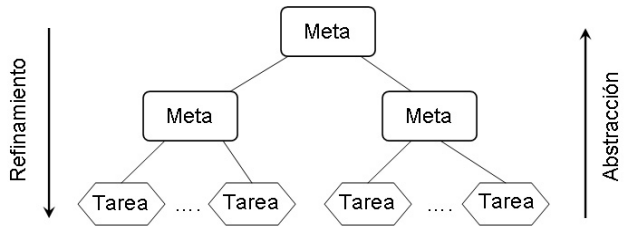


Figura 5.30. Estructura del ARM

En esta tesis, utilizamos la técnica de abstracción de metas para la construcción del ARM. Primero, se identifican los usuarios posteriormente las tareas de análisis y finalmente, se integran las tareas en metas de acuerdo a sus similitudes.

El proceso de integración seguido durante la construcción del ARM, permitirá reducir el número de tareas de tal forma, que sea más fácil de manejar y también permitirá eliminar tareas duplicadas.

El ARM resultante de la integración de tareas en metas, tendrá como raíz el proceso de negocio definido en el paso uno. El cual, representa la meta de análisis de más alto nivel.

Caso de estudio (continuación): en la figura 5.31, se muestra la descripción textual da las tareas de análisis relacionadas con el proceso de negocio *Ventas*.

La primera columna, describe a los usuarios del AD (*Gestor de Entregas, Gestor de Promociones y Gestor de Ventas*). En la segunda columna, se describen las tareas de análisis.

Usuario	*Tarea*
Gestor de Entregas	T1: analizar los canales utilizados para distribuir las mercancías con el fin de estimar cuál es el más eficiente.
	T2: analizar el servicio de entregas, con el fin de hacer un seguimiento de la calidad en las entregas de los artículos (cantidades entregadas demasiado pronto, cantidades entregadas demasiado tarde).
	T3: analizar el servicio de entregas de mercancías, con el fin de hacer un seguimiento del tiempo promedio de las entregas.

Usuario	*Tarea*
Gestor de Promociones	T4: analizar las ventas, con el fin de estudiar el desempeño en relación al número de órdenes de ventas generadas a partir de las promociones.
	T5: analizar las estrategias de publicidad de ventas según el tipo de público al que va dirigido (consumidor, comerciantes, etc.).
	T6: analizar el impacto que las promociones tienen sobre los artículos de promoción lenta.

Usuario	*Tarea*
Gestor de Ventas	T7: analizar las ventas, con el fin de determinar el número de productos vendidos de acuerdo a la presentación.
	T8: analizar las ventas, con el propósito de entender mejor el comportamiento del mercado en relación con las preferencias y tendencias de consumo.
	T9: analizar las ventas, con el fin de determinar las percepciones obtenidas en las distintas zonas geográficas.
	T10: analizar las ventas con el fin de comparar las ventas planeadas con las ventas actuales en relación al margen de ganancias y al margen de ganancias porcentual.
	T11: analizar las ventas, con la intención de estimar el impacto de las promociones en las tendencias de consumo, en relación al número de artículos vendidos.

Figura 5.31. Relación de tareas y usuarios

Una vez definidas las tareas, el siguiente paso para la construcción del ARM, consiste en integrarlas en una meta en común. Al realizar la integración de las tareas en metas en nuestro caso de estudio, identificamos tres metas:

Meta 1: Eficiencia y efectividad de las ventas (Esta meta integra las siguientes tareas: T7, T8, T9, T10, T11).

Meta 2: Impacto de las ofertas y promociones (Esta meta integra las siguientes tareas: T4, T5, T6).

Meta 3: Calidad del servicio de entrega (Esta meta integra las siguientes tareas: T1, T2, T3).

La figura 5.32, muestra el ARM resultante.

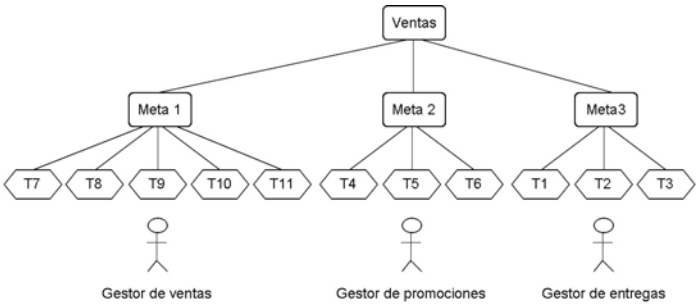


Figura 5.32. Representación gráfica del ARM

Siguiendo los lineamientos señalados en [Ins03] [VFP05], el ARM se convierte en el punto de partida para la construcción de los casos de uso y los diagramas de actividad ya que por cada meta se construirá un caso de uso y un diagrama de actividad.

Etapa 3: Especificación de los requisitos

Esta actividad, consiste en identificar los requisitos de usuario a partir de las metas identificadas en el ARM. Durante esta etapa, realizamos tres pasos: 1) descripción de las metas en casos de uso, 2) descripción de la interacción

del usuario con el sistema en diagramas de actividad y 3) descripción de los requisitos de información.

Paso 1. Descripción de las metas en casos de uso

Los casos de uso, permiten representar la funcionalidad del sistema y de aquí que hayan sido adoptados por numerosos métodos. Un caso de uso, representa la interacción entre el sistema y los actores que harán uso de la funcionalidad del sistema. Esta interacción, es usualmente dividida en frases de lenguaje natural (descripción del caso de uso). En la figura 5.33 y 5.34 se muestran la especificación en casos de uso para la *Meta 1* y *Meta 2* respectivamente.

Caso de Uso	Meta 1: Eficiencia y efectividad de las ventas
Actores	*Gestor de ventas*
Propósito	Permitir a los usuarios analizar la efectividad de las ventas.
Resumen	El análisis, consiste en el estudio de las ventas. Este análisis, proporcionará información al usuario sobre el margen de ganancias y el margen de ganancias porcentual. Algunas de las consultas que el usuario podrá realizar al interactuar con el sistema son: *Consultar el margen de ganancias ,Consultar el porcentaje de ganancias, Consultar el total de ventas.* El análisis podrá realizarse por producto, localización, fabricante y cliente.

Figura 5.33 Caso de uso relacionado con la meta 1

Caso de uso	Meta 2: Impacto de las ofertas y promociones
Actores	*Gestor de promociones*
Propósito	Permitir a los usuarios analizar los resultados de las ventas durante periodos de promociones.
Resumen	Este caso de uso, es iniciado por el usuario. Ofrece funcionalidad para el análisis de los resultados de las ventas en periodos promociónales. Alguna de las consultas que el usuario podrá realizar son: *Consultar el número de artículos vendidos durante periodos promociónales.* *Consultar el número de clientes que han realizado compras durante periodos promociónales.* *Consultar el número de órdenes generadas durante las promociones.* El análisis podrá realizarse por producto, fabricante y localización y tipo de promoción.

Figura 5.34. Caso de uso relacionado con la meta 2

Paso 2. Descripción de la interacción del usuario con el sistema en diagramas de actividad

Con la finalidad de describir la interacción existente entre los usuarios y el AD, utilizamos diagramas de actividad [OMG05a].

Esta descripción, parte de las acciones que el usuario, necesita realizar al interactuar con el AD para lograr con éxito cada una de las metas. Durante esta actividad, se realiza un diagrama de actividad por cada meta identificada en el ARM.

La figura 5.35, muestra el diagrama de actividad correspondientes al caso de uso: *Meta 2: Impacto de las ofertas y promociones.*

Cada actor participante, es representado por medio de una calle en la cual, se incluyen las actividades realizadas por cada uno de ellos. El diagrama, muestra los datos necesarios y producidos por cada actividad. Similar a [SLK05] [GOMNA00] los datos son representados como Elementos de Información (EI) que fluyen entre el usuario y el sistema. Durante la interacción se identifican dos tipos de EI:

EI de salida.- El sistema, proporciona información a los usuarios. Estos elementos, se usan para dar respuesta a una consulta.

EI de entrada.- El sistema, espera información por parte del usuario. Esta información, es utilizada por el sistema para realizar correctamente una actividad.

Los EI de salida y de entrada, se representan en el diagrama de actividad por medio de un cuadrado con el estereotipo <<Output>> e <<Input>> respectivamente.

El proceso de la figura 5.35, inicia cuando el usuario proporciona los datos a través de un EI tipo *Input* (*Fabricante, Localización, o Producto*).

A partir de esta información, el sistema realizará algunas de las siguientes acciones: *A) Consultar el número de artículos o B) Consultar el número de clientes*.

Una vez finalizada esta actividad, el sistema proporcionará al usuario el *Número de artículos* o el *Número de clientes* a través de un EI tipo *Output*.

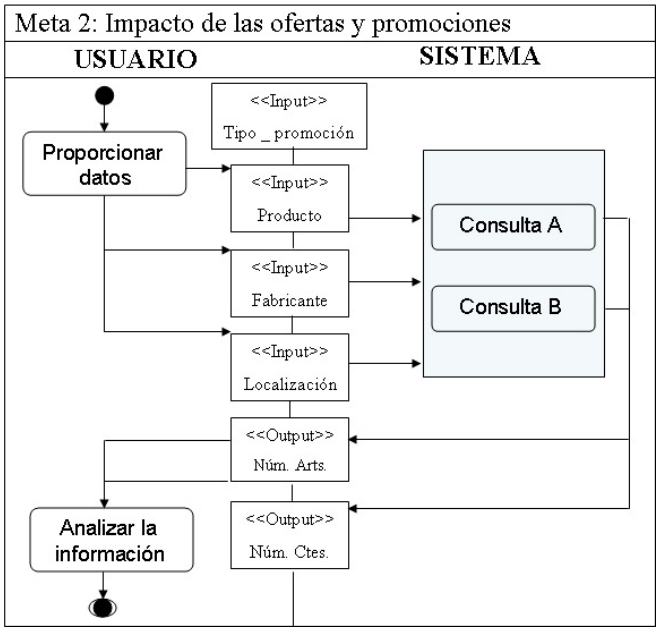


Figura 5.35. Diagrama de Actividad para la meta 2

En la figura 5.36, se muestran las actividades relacionadas con la *Meta 1: Eficiencia y efectividad de las ventas*.

El proceso, inicia cuando el usuario proporciona los datos *Localización, Fabricante, Cliente o Producto* (por medio de un EI tipo *Input*). A partir de esta información, el sistema podrá realizar algunas de las siguientes acciones: *A) Consultar el margen de ganancias, B) Consultar el porcentaje de ganancias o C) Consultar el total de ventas*. Dependiendo de la acción realizada, el sistema mostrará el *margen de ganancias, el porcentaje de ganancias o el total de ventas*.

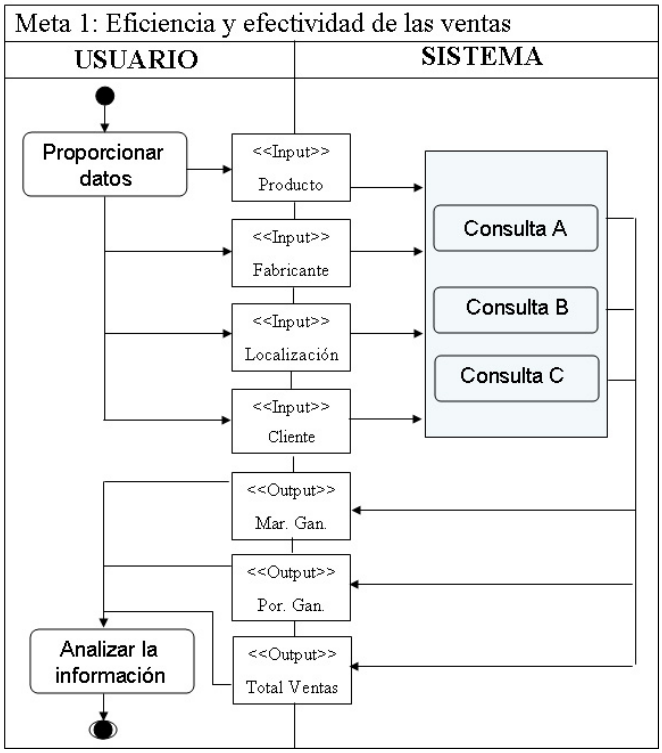


Figura 5.36. Diagrama de Actividad para la meta 1

Paso 3. Definición de los requisitos de información.

Los requisitos de información, representan los datos que serán almacenados en el AD y que se relacionan directamente con los conceptos del modelo multidimensional.

Para definir los requisitos de información, nosotros proponemos analizar cada diagrama de actividad y documentar los conceptos (dimensiones, niveles y medidas) que guiarán la selección y el refinamiento de un esquema multidimensional candidato. Por ejemplo, en la figura 5.38 se resumen los requisitos de información relacionados con las metas: *1) Eficiencia y efectividad de las ventas y 2) Impacto de las ofertas y promociones* (esta información, se obtuvo de los diagramas de actividad de la figura 5.35 y 5.36). En la figura 5.37, se muestra el nombre de la meta, el nombre de cada EI y el tipo (*Input, Output*). Esta información, debe ser relacionada con los conceptos del modelo multidimensional de la siguiente manera:

1. Cada EI tipo *Input*, debe ser relacionado con el concepto dimensión o nivel.
2. Cada EI tipo *Output*, debe ser relacionado con el concepto medida.

Caso de estudio (continuación): **s**iguiendo la guía definida previamente, la información necesaria para el análisis de los datos relacionados con la *Meta 1 y Meta 2* es:

Dimensiones: *Fabricante, Localización, Cliente y Producto.*

Medidas: *Margen de ganancias, Porcentaje de ganancias, Total de ventas, Número de artículos y Número de clientes.*

Meta	1: Eficiencia y efectividad de las ventas	
	Nombre	**EI**
	Margen de ganancias	OutPut
	Total de ventas	Output
	Porcentaje de ganancias	OutPut
	Fabricante	Input
	Cliente	Input
	Localización	Input
	Producto	Input

Requisitos de información para la meta 1

Meta	2: Impacto de las ofertas y promociones	
	Nombre	**EI**
	Número de artículos	Output
	Número de clientes	Output
	Localización	Input
	Producto	Input
	Fabricante	Input

Requisitos de información para la meta 2

Figura 5.37. Resumen de los requisitos de información

5.5. Fase 3. Integración.

Esta fase, consiste en unificar el conocimiento obtenido durante la fase 1 y la fase 2 con el objetivo de generar un esquema conceptual para el AD que permita capturar los requisitos de usuario y que este fuertemente acoplado con la base de datos operacional que nutrirá el AD.

Para lo cual, proponemos dos pasos:

1. *Selección*: en este paso, el esquema multidimensional candidato es seleccionado a partir de los requisitos de usuario.

2. *Refinamiento*: en este paso, el esquema multidimensional seleccionado es modificado manualmente.

5.5.1. Paso 1: Selección.

Para seleccionar un esquema multidimensional candidato, es necesario realizar una comparación entre los elementos que componen la estructura del esquema candidato (dimensiones, niveles y medidas) y los requisitos de información (EI *Input* y EI *Output*).

El resultado de esta comparación, es un conjunto de relaciones semánticas.

Este conjunto de relaciones semánticas, ofrecerá al diseñador el conocimiento necesario para elegir un esquema candidato del conjunto de esquemas generados en la fase 1.

Durante este paso, utilizamos una técnica de comparación estructural.

Esta técnica, consiste en definir las relaciones semánticas entre los elementos de dos esquemas distintos sin considerar sus posibles instancias [RB01]. Las relaciones semánticas que pueden identificarse son:

- *Directa:* una relación será considera directa, si dos elementos individuales de distintos esquemas se corresponden semánticamente entre sí.

  Para lograr esta correspondencia, no es necesario introducir ninguna formula o expresión de correspondencia adicional.

- *Indirecta:* una relación será considerada indirecta, si dos o más elementos individuales de un esquema se corresponden semánticamente con algún elemento individual de otro esquema.

  Para establecer esta relación, es necesario introducir formulas o expresiones de correspondencia adicional.

Por ejemplo, en la tabla 5.1, se muestra que el atributo *Precio* del esquema A y el atributo *Importe* del esquema B tienen una relación semántica directa.

También, se muestra que el atributo *Coste* del esquema B tiene una relación semántica indirecta con los atributos *Precio e Impuesto* del esquema A.

Por lo tanto, es necesario introducir una expresión de correspondencia *(Coste = Precio * (1+Impuesto/100))* para lograr una relación semántica correcta.

Tabla 5.1. Correspondencias directas e indirectas

Esquema A	*Esquema B*	*Expresión de Correspondencia*	*Correspon-Dencia*
Precio	Importe	Precio = Importe	Directa
Precio, Impuesto	Coste	Coste = Precio * (1+Impuesto/100)	Indirecta

Para seleccionar un esquema multidimensional candidato, es necesario precisar todas las relaciones semánticas (directas e indirectas) entre los elementos del esquema (dimensiones, niveles y medidas) y los Elementos de Información (EI) identificados durante la fase 2.

Este proceso, se lleva a cabo de la siguiente manera:

- Cada EI tipo *Output*, puede ser relacionado semánticamente con las medidas de la clase de cubo, o bien con algún atributo que permita establecer correctamente la relación de correspondencia semántica.
- Cada EI tipo *Input*, puede ser relacionado semánticamente con las dimensiones o con los niveles del esquema multidimensional candidato.

Ejemplo (Continuación). En la tabla 5.2, se muestra el conjunto de relaciones semánticas identificadas entre los elementos del esquema candidato *Línea* (figura 5.29(a)) y los EI tipo *Output* identificados en la fase 2 (figura 5.37).

En la primera columna de la tabla, se detallan los elementos del esquema candidato que participan en la evaluación.

En la segunda, se muestran los requisitos de información. La expresión de correspondencia, se muestra en la tercera columna y por último se indica el tipo de relación semántica.

Tabla 5.2. Relaciones semánticas entre los EI *Output* y *Línea*

Elementos de Línea	**EI Output**	**Expresión de correspondencia**	**Tipo de correspondencia**
Cantidad, Precio	Total de ventas	Sum (cantidad * precio)	Indirecta
Cantidad, Precio, Precio_de_coste	Margen de ganancias	Sum(cantidad * precio) – Sum (cantidad * precio_coste)	Indirecta
Cantidad	Número de artículos	Sum (cantidad)	Indirecta
Cliente	Número de clientes	Count (Cliente)	Indirecta
cantidad, precio, precio_de_coste	Porcentaje de ganancias	Margen de ganancias / Sum (cantidad * precio_coste) *100	Indirecta

Como se muestra en la tabla 5.2, todas las correspondencias identificadas son indirectas. Por lo tanto, es necesario introducir una expresión de correspondencia que permita establecer la semántica correcta entre los elementos.

Por ejemplo, para establecer la relación semántica entre el EI *Total de ventas* y los elementos *cantidad* y *precio* del esquema candidato *Linea* es necesario introducir la expresión de correspondencia *Sum (cantidad * precio).*

Esta correspondencia, le indica al diseñador que a partir de dos elementos del esquema multidimensional candidato es posible satisfacer un requisito de información. También, le indica que el *Total de ventas* deberá ser representado en el esquema seleccionado por medio de la formula *Sum (cantidad * precio).*

En la tabla 5.3, se muestran las relaciones semánticas identificadas entre los EI tipo *Input* y los elementos del esquema candidato *Línea*. Como se pude ver, todas las relaciones semánticas identificadas son directas.

Tabla 5.3. Relaciones semánticas entre EI *Input* y CLínea

Elementos de CLínea	EI Input	Expresión de correspondencia	Tipo de correspondencia
Artículo	Producto	Producto = Artículo	Directa
Fabricante	Fabricante	Fabricante = Fabricante	Directa
Ciudad	Localización	Localización = Ciudad	Directa
Cliente	Cliente	Cliente = Cliente	Directa

El EI *Producto,* se relaciona semánticamente con el elemento *Artículo* del esquema candidato *Línea* sin necesidad de introducir ninguna expresión de correspondencia adicional.

Esta correspondencia directa, le indica al diseñador que es posible satisfacer el requisito de información *Producto* a partir del elemento *Artículo.* También, le indica que el requisito de información *Producto* será representado en el esquema multidimensional seleccionado a través de la clase *Artículo.*

En la tabla 5.4, se resume el número de correspondencias semánticas (directas e indirectas) identificadas entre los requisitos de información y los elementos de los esquemas candidatos *Línea* y *Pronósticos*.

Tabla 5.4. Resumen de correspondencias semánticas

Elemento\Esquema candidato	Línea	Pronósticos
EI Input	4	2
EI Output	5	0

En la tabla 5.4, se muestra que cuatro elementos del esquema *Línea*, se relacionan semánticamente con los EI tipo *Input* mientras que sólo dos elementos del esquema *Pronósticos* se relacionan semánticamente con estos elementos.

También, se muestra que cinco elementos del esquema *Línea* se relacionan con los EI tipo *Output* y que ningún elemento de *Pronósticos* se relaciona con estos elementos.

A partir de esta información, es posible seleccionar el esquema candidato *Línea* ya que es el que mejor refleja los requisitos de usuario y además garantiza la disponibilidad de los datos en la base de datos operacional.

Durante este paso, es posible que dos esquemas candidatos cumplan con el mismo número de requisitos de usuario. En este caso, es necesario establecer prioridades en los requisitos de información y de esta forma, seleccionar el esquema multidimensional candidato más apropiado.

5.5.2. Paso 2: Refinamiento manual.

Una vez seleccionado el esquema multidimensional, éste debe ser modificado manualmente con el fin de adecuarlo a los requisitos de usuario.

Similar a [GMR98b] algunos de los cambios que pueden ser realizados al esquema multidimensional seleccionado son: a*ñadir medidas, agregar la dimensión tiempo, eliminar niveles de dimensión y agregar restricciones de aditividad*.

- ***Añadir medidas.-*** Las expresiones de las correspondencias indirectas para los EI tipo *Output*, deben ser añadidas a la clase de cubo del esquema multidimensional seleccionado.

  Caso de estudio (Continuación)*:* Las siguientes medidas deben añadirse a la clase de cubo del esquema *Línea*.

  Total de ventas = Sum (cantidad *precio)
  Margen de ganancias = Sum (cantidad *precio)-
  Sum (cantidad*precio_coste)
  Número de artículos = Sum (cantidad)
  Número de clientes = Count (Cliente)
  Porcentaje de ganancias = Margen de ganancias /
  Sum (cantidad*precio_coste)*100

- ***Añadir la dimensión tiempo.***- Ya que el análisis de los datos puede ser realizado en diferentes periodos de

tiempo, es necesario añadir la dimensión tiempo al esquema multidimensional seleccionado.

Esta dimensión, podrá ser derivada del esquema multidimensional seleccionado si existe en él algún atributo tipo fecha.

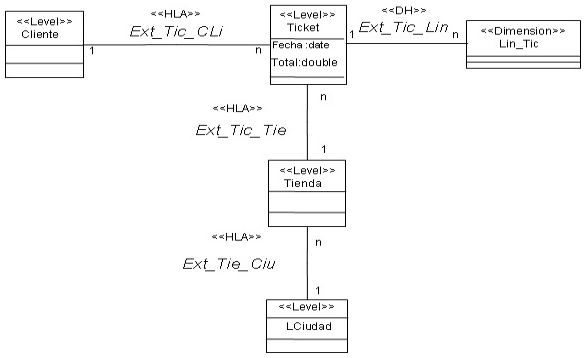
Caso de estudio (Continuación): *La dimensión* Tiempo *del esquema* Línea*,* puede generarse a partir del atributo Fecha. Este atributo se localiza en el nivel raíz (Ticket) de la dimensión Lin_Tic*.*

El atributo Fecha, debe ser analizado para determinar los niveles de la dimensión Tiempo. En nuestro caso de estudio, podemos asumir que los niveles que forman a la dimensión Tiempo son: Día→Mes→Año.

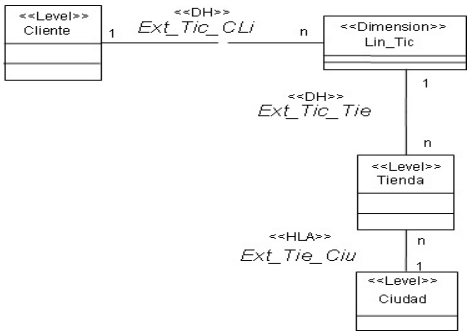
— ***Eliminar niveles de dimensión.-*** Posiblemente, no todos los niveles del esquema multidimensional seleccionado sean de interés para el análisis de los datos, por lo tanto estos niveles deben ser eliminados.

Caso de estudio (Continuación): En la figura 5.38(a), se muestra el subesquema del esquema seleccionado *Línea*. El nivel raíz *Ticket* de la dimensión *Lin_Tic* (figura 5.38(a)) puede ser eliminado ya que la información que éste contiene (*Total*) puede obtenerse a partir de la información de la clase de cubo *Línea* (*precio_unitario, cantidad*).

Al eliminar el nivel raíz, sus descendientes (*Cliente, Tienda y Ciudad*), se añaden como niveles de la dimensión (figura 5.38(b)).



a) Sub-esquema original



b) Sub-esquema reestructurado

Figura. 5.38. Dimensión *Lin_Tic* reestructurada

— ***Añadir restricciones de aditividad.-*** Una vez definidas las medidas y las dimensiones, es necesario clasificarlas como: aditivas, semiaditivas y no aditivas.

Con el fin de clasificar las medidas, emplearemos la categorización propuesta en [HSC04]. En la tabla 5.5, se muestra que las medidas cuyo valor resulte de

aplicar una función de agregación Avg, Max o Min, se consideran no aditivas. Mientras que las medidas que son resultado de valores que se repiten en un periodo de tiempo son consideradas semiaditivas.

Tabla 5.5. Clasificación de medidas

Categoría no aditiva		**Característica**
Fracciones	Frecuencias	Medidas que incluyen operaciones con numerador y denominador
	Porcentajes	
Promedio (Avg)		Funciones de agregación distinta a SUM y COUNT
Máximos(Max), Mínimos(Min)		
Categoría semiaditiva		**Característica**
Periodo		Se repiten valores sobre el tiempo, por lo que al sumar se duplican esos valores.
Categoría		Se repiten valores sobre diferentes tipos de instancias.

Caso de estudio (Continuación).- *Siguiendo esta clasificación identificamos las siguientes restricciones de aditividad:*

- Número de clientes *es semiaditiva sobre la dimensión* Lin_Art.

- Margen de ganancias *y* Porcentaje de ganancias *son no aditivas.*

Similar a [TPGS01] las restricciones de aditividad, las representaremos en el esquema multidimensional como etiquetas UML. En la figura 5.39, se muestra el esquema multidimensional reestructurado. Los cambios realizados al esquema son:

- Se añadieron las expresiones de correspondencias indirectas a la clase de cubo Línea. Por ejemplo, se añadió la expresión de correspondencia Total_ventas.
- Se añadió la dimensión Tiempo (asumiendo que el granulo es Día→Mes→Año).
- Se elimino el nodo raíz de la dimensión Lin_Tic.
- Se añadieron las restricciones de aditividad usando etiquetas UML.
- Los EI que originaron relaciones de correspondencias semánticas directas, se representaron por medio de su equivalente semántico del esquema candidato. Por ejemplo, el requisito de información Producto es representado en el esquema seleccionado por medio del nivel Artículo.

{Margen de ganancias y porcentaje de ganancias son medidas no aditivas}
{Número de clientes es medida semiaditiva sobre la dimension Lin_Art}

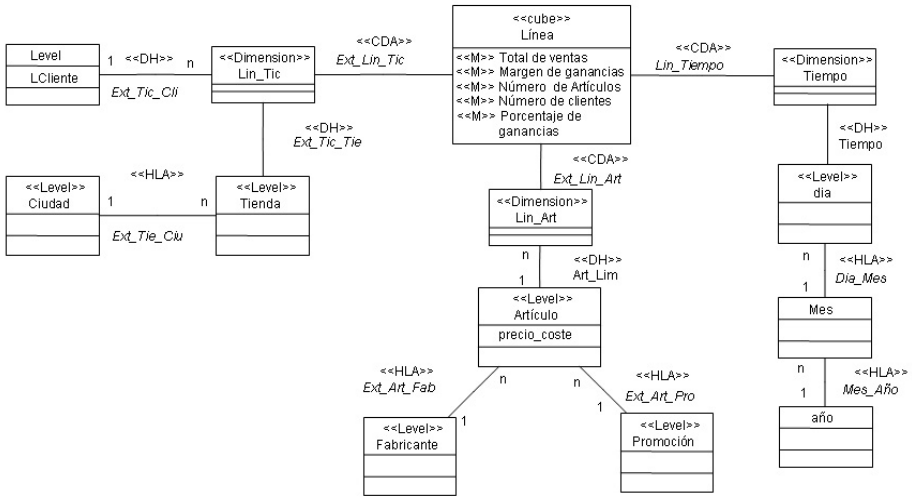


Figura 5.39. Esquema multidimensional final

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo, se presentan las conclusiones a las que se han llegado con el desarrollo de esta tesis, se detallan las contribuciones más importantes que se han realizado, las publicaciones derivadas de este trabajo de investigación y por último los trabajos futuros previstos.

6.1. Conclusiones

En el capítulo 1, se propuso como objetivo principal de esta tesis la definición de una metodología para el diseño conceptual de ADs, que integrara dos aproximaciones existentes: análisis del esquema conceptual de las bases de datos operacionales y análisis de los requisitos de usuario.

La metodología que se ha propuesto en este trabajo (capítulo 5), se estructura en tres fases.

La primera fase, consiste en la derivación semiautomática del esquema conceptual del AD a partir de los esquemas conceptuales de las bases de datos operacionales. Para ello, se han definido un conjunto de reglas de transformación a nivel metamodelo en el marco de MDA. El conjunto de reglas de transformación muestra una relación semántica entre los conceptos del metamodelo ER y los conceptos del metamodelo OLAP. El trabajo, demuestra que las ideas de MDA son perfectamente válidas para el diseño de ADs. El uso de MDA en el diseño conceptual de ADs se justifica por la necesidad de emplear un marco de desarrollo estandarizado.

En la segunda fase, se define un modelo de requisitos de usuario. El análisis de requisitos, es un proceso difícil en el cual la información se obtiene a partir de distintas fuentes. El modelo de requisitos que se propone, en esta tesis se obtiene a través de un proceso iterativo, utilizando un método basado en metas El objetivo de este método, es identificar las metas y las tareas necesarias para desempeñar el proceso de negocio. Para mostrar la relación entre las metas, las tareas y los usuarios empleamos un Árbol de

Refinamiento de Metas (ARM). La creación del ARM, puede ser realizada empleando diferentes técnicas, en esta tesis se utiliza la técnica de identificación de metas por abstracción ya que es mas fácil identificar cada una de las tareas que realiza el usuario y posteriormente integrarlas a una meta especifica.

En la tercera fase, fase de integración, los requisitos de usuario obtenidos en la segunda fase son contrastados con los esquemas multidimensionales candidatos obtenidos en la primera fase. El objetivo, es seleccionar el esquema multidimensional que mejor satisfaga los requisitos de usuario. Esta fase se lleva a cabo en dos pasos: selección y refinamiento manual. La selección se realiza aplicando una técnica de comparación estructural, que permite establecer una relación semántica entre los elementos de los esquemas candidatos y los requisitos de usuario.

A continuación se resumen las ventajas derivadas de nuestra metodología:

1. El esquema multidimensional final, se encuentra fuertemente respaldado con la base de datos operacional, por lo que el proceso de Extracción, Transformación y Carga se facilita.
2. La integración de la fase 1 y la fase 2, permite contrastar de manera coherente los requisitos de información de los usuarios y la disponibilidad de esta información en las bases de datos operacionales.

3. El empleo de una notación estandarizada para representar el esquema multidimensional, facilita la comprensión y la construcción del AD.

6.2. Contribuciones.

En general, la mayoría de los trabajos existentes sobre diseño conceptual de ADs no relacionan el análisis de requisitos con el análisis del esquema conceptual de la base de datos operacional.

Los trabajos ([GRG05], [BCCFP01]) que plantean una relación entre éstas dos actividades, emplean métodos y notaciones no estandarizados. La contribución principal de este trabajo, consiste en una metodología que considera los requisitos de usuario y la disponibilidad de los datos en la base de datos operacional.

Esta metodología, se basa en los estándares de MDA. Tanto las reglas de transformación y el proceso de integración propuesto, constituyen un paso adelante en la sistematización del diseño conceptual de ADs.

En resumen, las contribuciones más relevantes de esta tesis son las siguientes:

1. La definición de un conjunto de reglas de transformación (en el marco de MDA) entre el metamodelo ER y el metamodelo OLAP como punto de partida para el diseño conceptual del AD.

2. La implementación del conjunto de reglas de transformación que dan soporte a nuestra metodología.
3. La definición de un método para identificar el conjunto de esquemas multidimensionales implícitos en el esquema conceptual de la base de datos operacional, utilizando las reglas de transformación definidas.
4. La definición de un método basado en metas para identificar los requisitos de usuario. Este método integra tres técnicas complementarias para la definición de los requisitos de usuario.
5. La integración final de estos dos métodos en una metodología única para el diseño conceptual de ADs.

6.3. Publicaciones.

Como resultado de este trabajo de investigación, se han realizado las siguientes publicaciones:

1. [ZC03] Zepeda L., Celma M. **Metodología para el Diseño Conceptual de Almacenes de Datos**. En *Proceedings of XII Congreso Internacional de Computación (CIC'03),* pages 22 - 29. México, Noviembre, 2003
2. [ZC04] Zepeda L., Celma M. **Perfil para el Diseño Conceptual de Almacenes de Datos**. En *Proceedings of XIII Congreso Internacional de*

Computación (CIC'04), pages 43 – 51. México, Noviembre 2004.

3. [ZC05a] Zepeda L., Celma M. **Specifying Metamodel Transformations for Data Warehouse Design**. En *43rd ACM Southeast Conference (ACMSE'05),* pages 266-267. Atlanta, Georgia, USA, March 18-20, 2005. ACM.
4. [ZC05b] Zepeda L., Celma M. **A Methodology Framework for Conceptual Data Warehouse Design**. En *43rd ACM Southeast Conference (ACMSE'05),* pages 256-259. Atlanta, Georgia, USA, March 18-20, 2005. ACM.
5. [ZC05c] Zepeda L., Celma M. **Metodología de Diseño de Esquemas Multidimensionales basada en la Transformación de Modelos y Requisitos de Usuario**. En *8º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software, (IDEAS'05)*. Valparaíso, Chile, Mayo 02-06, 2005.
6. [ZC06a] Zepeda L., Celma M. **Especificación de Requisitos para el Diseño Conceptual de Almacenes de Datos.** En *V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'06)*, pages 117-124. Puebla, México, Febrero 1-3, 2006.
7. [ZC06b] Zepeda L., Celma M. **Aplicando MDA al Diseño Conceptual de Almacenes de Datos**. En *V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento*

(JIISIC'06), pages 271-278. Puebla, México, Febrero 1-3, 2006.

8. [ZC06c] Zepeda L., Celma M. **A Model Driven Approach for Data Warehouse Conceptual Design**. En *7th International Baltic Conference on Databases and Information Systems (DB&IS'06).* Vilnius, Lithuania, July 04 - 06, 2006. IEEE Computer Society.
9. [ZC06d] Zepeda L., Celma M. **Diseño Conceptual de Almacenes de Datos con MDA**. En *9º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS'06).* Mar del Plata, Argentina, Abril 24-28, 2006.
10. [ZC06e] Zepeda L., Celma M. **Applying MDA to the Conceptual Design of Data Warehouses**. *En Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, pages 156-161. San Francisco, CA, USA, July 5-7, 2006.
11. [ZC06f] Zepeda L., Celma M. **Conceptual Design of Data Warehouses with MDA**. En *Advances in Databases and Information Systems, 10th East European Conference, (ADBIS'06).* Thessaloniki, Greece, September 3-7, 2006.
12. [ZC08a] Zepeda L., Celma M. **A Goal Method for Conceptual Data Warehouse Design.** *En 10th International Conference on Enterprise Information Systems (ICEIS'08).* Barcelona, España, June 12-16, 2008

13. [ZC08b] Zepeda L, Celma M. **A Mixed Approach for Data Warehouse Conceptual Design with MDA**. En *The International Conference on Computational Science and Its Applications (ICCSA'08).* Perugia, Italia, June 30th - July 3rd, 2008.

6.4. Trabajos futuros.

El trabajo realizado en esta tesis, puede ser continuado siguiendo diferentes líneas de investigación como son:

- Es bien sabido que un AD puede ser implementado empleando tecnología ROLAP o tecnología MOLAP. Una línea de trabajo futura puede ser la definición e implementación de un conjunto de reglas de transformación del esquema conceptual del AD a un esquema lógico (ROLAP o MOLAP).
- Siguiendo la propuesta de MDA para la generación de código a partir de modelos, una evolución de este trabajo puede ser también la definición e implementación de un conjunto de reglas de transformación a partir del esquema lógico del AD a código.
- Otra línea de trabajo posible consistiría en el estudio y propuesta de un modelo de calidad para los requisitos de usuario del AD. Este modelo de calidad podría basarse en el estándar ISO 9126 -1, el cual clasifica la calidad

del software en un conjunto de características tales como: funcionalidad, usabilidad, mantenimiento y eficiencia.

— El objetivo de las metodologías de modelado conceptual, consiste en proporcionar un medio que permitan generar esquemas representativos y fiables de la realidad. Otro trabajo futuro, puede ser la definición de un conjunto de métricas de calidad para esquemas conceptuales de ADs tales como: facilidad de comprensión, simplicidad, implementabilidad, integración, autoexplicación, minimilidad de información y completitud de información entre otros.

Bibliografía

[Ant96] Antón A. Goal Based Requirements Analysis. *In Proceedings of 2nd International Conference on Requirements Engineering (ICRE '96)*, pages 136 – 144, Colorado Springs, Colorado, USA, April 15 – 18 1996. IEEE Computer Society.

[AVK06] Stahl T., Völter M., Krzysztof C. *Model-Driven Software Development: Technology, Engineering, Management.* Ed. WILEY, 3rd Edition, 2006.

[BCCFP01] Bonifati A., Cattaneo F., Ceri S., Fuggetta A., Paraboschi S. Designing Data Marts for Data Warehouses. *ACM Transactions on Software Engineering and Methodology*, 10(4):452 - 483, 2001.

[BGGMP04] Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J., Perini A. Tropos: An Agent-oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems,* 8(3):203 – 236, 2004.

[BNRH88] Burton A. M., Nigel Shadbolt, Rugg G., Hedgecock A. Knowledge Elicitation Techniques in Classification Domains. *In Proceedings of the 8th European Conference on Artificial Intelligence (ECAI'88)*, pages 85– 90, Munich, Germany, August 1–5, 1988.

[Bor06] Borland Together Architect 2006. Internet: http://www.borland.com/downloads/download_together.html, 2006.

[CPP76] Chen Peter P. The Entity Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1): 9 – 36, 1976.

[CT98] Cabibbo L., Torlone R. A Logical Approach to Multidimensional Databases. *In Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, Volume 1377 of *Lecture Notes in Computer Science*, pages 183 – 197, Valencia, Spain, March 23 - 27 1998. Springer-Verlag.

[EPMT04] Estrada H., Pastor O., Martinez A., Torres-J. Using a Goal-Refinement Tree to Obtain and Refine Organizational Requirements. *In Proceedings of the Computational Science and its Applications (ICCSA'04)*, Volume 3046 of *Lecture Notes in Computer Science*, pages 506-513, Assisi, Italy, May 14-17 2004. Springer-Verlag.

[GMR98a] Golfarelli M., Maio D., Rizzi S. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *International Journal of Cooperative Information System,* 7 (2-3): 215 – 247, 1998.

[GMR98b] Golfarelli M., Maio D., Rizzi S. Conceptual Design of Data Warehousing from E/R Schemes. *In Proceedings of the 31st Hawaii International Conference on Systems Sciences (HICSS'98)*, Volume 7, pages 334 – 343, Kohala Coast, Hawaii, USA, January 6 - 9 1998. IEEE Computer Society.

[GOMNA00] García Molina J., Ortin-I. M., Moros B., Nicolás J., Álvarez J. Toward Use Case and Conceptual Models through Business Modeling. *In Proceedings of the 19th International Conference on Conceptual Modeling (ER'00)*, Volume 1920 of *Lecture Notes in Computer Science*, pages 281-294, Salt Lake City, Utah, USA, October 9 – 12 2000. Springer-Verlag.

[GRG05] Giorgini P., Rizzi S., Garzetti M. Goal-Oriented Requirements Analysis for Data Warehouse Design. *In Proceedings of the ACM 8th International Workshop on Data Warehousing and OLAP (DOLAP'05)*, pages 47 – 56, Bremen, Germany, November 4 - 5 2005. ACM.

[HSC04] Horner J., Song I-Y., Chen Peter P. An Analysis of Additivity in OLAP Systems. *In Proceedings of the ACM 7th International Workshop on Data Warehousing and OLAP (DOLAP'04)*, pages 12 – 13, Washington, DC, USA November 12 - 13 2004.

[Inm02] Inmon W.H. *Building the Data Warehouse.* John Wiley & Sons, 3rd edition, 2002.

[Ins03] Insfrán, E. *A Requirements Engineering Approach for Object-Oriented Conceptual Modeling*. Tesis de Doctorado. Departamento de sistemas informáticos y computación. Universidad politécnica de Valencia, España Octubre 2003.

[KR02] Kimball, R., Ross, M. *The Data Warehouse Toolkit*. John Wiley & Sons, Second edition, 2002.

[KWB03] Kleppe A., Warmer J., Bast W. *MDA Explained. The Practice and Promise of the Model Driven Architecture*. Addison Wesley, First edition, 2003.

[Lam01] Lamsweerde A. Goal-Oriented Requirements Engineering: A Guided Tour, *Proc. 5º IEEE International Symposium on Requirements Engineering.* Toronto, Canadá, 2001.

[LS97] Lenz H., Shoshani A. Summarizability in OLAP and Statical Data Bases. *In Proceedings of 9th International Conference on Scientific and Statistical Database Management (SSDBM'97),* pages 132 – 143, Olympia, Washington, USA, August 132 – 143 1997. IEEE Computer Society.

[LTS02a] Luján-Mora S., Trujillo J., Song I-Y. Multidimensional modeling with UML package diagrams. *In Proceedings of the 21st International Conference on Conceptual Modeling (ER'02)*, Volume 2503 of *Lecture Notes in Computer Science*, pages 199 – 213, Tampere, Finland, October 7 – 11 2002. Springer-Verlag.

[LTS02b] Luján-Mora S., Trujillo J., Song I-Y. Extending UML for Multidimensional Modeling. *In Proceedings of the 5th International Conference on the Unified Modeling Language (UML'02)*, Volume 2460 of *Lecture Notes in Computer Science*, pages 290-304, Dresden, Germany, September 30 - October 4 2002. Springer-Verlag.

[MP04] Molina JC., Pastor O. *MDA,* OO-Method y la Tecnología OLIVANOVA. Model Execution*. Internet:* http: www.dsic.upv.es/workshops/dsdm04/files/08-Molina.pdf, 2004.

[MSUW04] Mellor S., Scott K., Uhl A., Weise D. *MDA Distilled: Principles of Model-Driven Architecture*. Addison-Wesley, Second edition, 2004.

[MTSP05] Mazón J-N, Trujillo J., Serrano M., Piattini M. Designing Data Warehouses: From Business Requirement Analysis to Multidimensional Modeling. *In Proceedings of the 1st Int. Workshop on Requirements Engineering for Business Need and IT Alignment*, Paris, France, August 29 – September 2, 2005.

[OMG03a] Object Management Group (OMG). *Common Warehouse Metamodel 1.1 (Volume 1) (CWM 1.1).* Internet: http://www.omg.org/docs/formal/03-03-02.pdf, March 2003.

[OMG03b] OMG*. Common Warehouse Metamodel 1.0.1 (Volume 2) (CWM 1.0.1) Specification*. Internet: http://www.omg.org/cgi-bin/doc?formal/03-03-02, March 2003*.*

[OMG04] OMG. Q*VT-Partners, revised submission for MOF 2.0 Query/View/Transformations (QVT) RFP*. Internet: http://www.omg.org/docs/ad/04-04-01.pdf, April 2004.

[OMG05a] OMG. *Unified Modeling Language 2.0 Specification (UML 2.0*). Internet: http://www.omg.org/docs/formal/05-07-05.pdf, July 2005.

[OMG05b] OMG. *Second Revised Submission: Meta Object Facility 2.0 (MOF 2.0) QVT*. Internet: http://www.omg.org/docs/ptc/05-11-01.pdf, November 2005.

[OMG06a] OMG. *Model Driven Architecture 2.0 (MDA 2.0) Core Specification.* Internet: http://www.omg.org/docs/formal/06-01-01.pdf, January 2006.

[OMG06b] OMG. *UML 2.0 Object Constraint Language (OCL) Specification*. Internet: http://www.omg.org/docs/formal/06-05-01.pdf, May 2006.

[PD02] Phipps C., Davis K. Automating Data Warehouse Conceptual Schema Design and Evolution. *In Proceedings of the 4th International Workshop Design and Management of Data Warehouses (DMDW'02)*, pages 23 – 32, Toronto, May 2002.

[PJD01] Pedersen T., Jensen Ch., Dyreson C. A foundation for Capturing and Quering Complex Multidimensional Data. *Information Systems*, 26(5): 383-423, 2001.

[RB01] Rahm E., A. Bernstein P. A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases*, 10 (4): 334 – 350, 2001.

[SLK05] Stefanov V., List B.,Korherr B. Extending UML 2 Activity Diagram with Business Intelligence Objects*. In Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'05)*, Volume 3589 of *Lecture Notes in Computer Science*, pages 53 – 63, Copenhagen, Denmark, August 22 - 26 2005. Springer-Verlag.

[SRME01] Song I-Y., Rowen W., Medsker Carl, Ewen Edward F. An Analysis of Many-to-Many Relationships Between Fact and Dimension Tables in Dimensional Modeling. *In Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses (DMDW'01)*, Interlaken, Switzerland, June 4 2001

[TPGS01] Trujillo J-C., Palomar M., Gómez J., Song I. Designing Data Warehouses with OO Conceptual Models*. IEEE Computer,* 34 (12): 66 – 75, 2001.

[VFP05] Valderas P., Fons J, Pelechano P. Using Task Descriptions for the Specification of Web Application Requirements. *Workshop en Ingeniería de Requisitos (WER),* pages 257-268. Porto, Portugal, Junio 13-14, 2005.

[Wes01] Westerman P. *Data Warehousing: using the Wal-Mart Model*. Morgan Kaufmann, First edition, 2001.

Anexo A

Programación en QVT

En éste anexo, se presenta la implementación de las reglas de transformación descritas en el capítulo 5 utilizando el lenguaje QVT[10].

[10] *La sintaxis que utilizamos en este anexo, es la soportada por Borland Together Architect 2006 [Bor06].*

A.1. Lenguaje operacional de QVT.

La parte operacional o imperativa de la especificación de QVT, se compone de un lenguaje operacional. El lenguaje operacional, se define como un medio estandarizado para la implementación de las reglas de transformación utilizando construcciones imperativas. Este lenguaje, se compone por extensiones de OCL y permite la descripción de las transformaciones siguiendo un estilo procedural. Las transformaciones escritas con este lenguaje, son llamadas transformaciones operacionales.

A.1.1. Transformación Operacional.

Una transformación operacional, representa la definición de una transformación unidireccional expresada en términos imperativos. Una transformación operacional, se compone por un conjunto de métodos (*Mappings* o *Queries*) y un método de inicio (*Main*) para la ejecución de la transformación.

Mapping.- Un *Mapping*, es un método usado para crear un nuevo elemento en un modelo y puede ser usado para invocar otros *Mappings*. Un *Mapping,* se compone de una signatura, un cuerpo y las cláusulas opcionales *When* e *Init*.

La signatura, define el contexto del *Mapping* y el tipo de dato de salida. La cláusula *When*, especifica una condición booleana. El *Mapping*, será realizado sólo si esta condición es verdadera. El cuerpo del *Mapping*, consiste de una sección de inicialización (*Init*) opcional en la cual se declaran e inicializadas variables y de la sección *Object* donde se forma el resultado del *Mapping* (figura A.1).

```
Mapping <Contexto> :: <nombre> ( [lista de parámetros]):
dato de salida
[When { código }]
{ [Init { código }]
Object { código }
}
```

Figura A.1. Cuerpo del Mapping

Main.- El método *Main*, es un *Mapping* especial a partir del cual inicia el proceso de transformación. Este método, se forma por un parámetro de entrada asociado a un metamodelo fuente y devuelve como resultado una instancia asociada a un metamodelo destino (figura A.2).

El resultado, al igual que en el *Mapping* se forma en la sección *Object*. La definición de los metamodelos que participan en la transformación, se realiza con la sentencia *metamodel,* que anteceden al método *Main*.

```
transformation [paquete.]<nombre>


metamodel  <metamodelo_fuente>;
metamodel  <metamodelo_destino>;

mapping main (in <nombre:> <metamodelo_fuente>):
                <nombre:><metamodelo_destino>
{
 object { <expresiones OCL>}
}
```

Figura A.2. Método Main

Query.- Un *Query*, es un método usado para realizar consultas sobre un modelo. Un *Query*, se compone por una

signatura y un cuerpo. El cuerpo del *Query*, se forma por una lista de expresiones separadas por un punto y coma (figura A.3). El *Query,* devuelve como resultado el valor de la última expresión OCL evaluada.

```
Query <Contexto> :: <nombre> ( [lista de parámetros]):
dato de salida
{
 Expresión A;
    ……
 Expresión B
}
```

Figura A.3. Cuerpo de un Query

A.2. Descripción de las Reglas de Transformación.

En esta sección se describen las reglas de transformación.

Main.- El *Mapping main*, representa la regla principal del proceso de transformación entre un modelo ER y un modelo OLAP (figura A.4).

En este *Mapping* se realizan las siguientes acciones:

1. Se identifican las entidades de cubo candidatas. Lo cual, se logra por medio del *Query Iscandidata().* El resultado de este *Query*, se almacena en la variable *ent_candidatas* (línea 6).

2. Se identifican los roles de cada relación que será trasformada en una dimensión. Lo cual, se logra por medio del *Query ext_rel_can()*. El resultado de este *Query*, se almacena en la variable *rels* (línea 7).
3. Se transforman las relaciones en dimensiones. Lo cual, se logra por medio del *Mapping RelationShipToDimension()* (línea 10).
4. Se transforma cada entidad de cubo candidata en un cubo. Lo cual, se logra por medio del *Mapping EntityToCube()* (línea 11).

Los cubos, son creados por el *Mapping EntityToCube()*. Para lo cual, se realiza una iteración sobre los datos almacenados en la variable *ent_candidatas* y se crea un cubo por cada dato almacenado en ella.

La regla de transformación *RelationShipToDimension()*, sigue el mismo procedimiento para crear las dimensiones sólo que la iteración se realiza sobre las datos almacenados en la variable *rels.*

AttributeToMeasure.- El *Mapping AttributeToMeasure*() (figura A.6), transforma un atributo en una medida (línea 1). El nombre de la medida, se compone por el nombre del atributo al cual se le ha concatenado el carácter *´M´* (línea 3).

```
1. transformation EntRel_To_Olap;
2. metamodel 'http://EntRel';
3. metamodel 'http://Olap';

4. mapping main (in er: EntRel::Schema) : Olap::Schema
   {
5. init
    {
6.   var ent_candidatas: Set(EntRel:Entity):=
     Iscandidata(er);
7.   var rels:Set (EntRel:RelationShipEnd):=
     ext_rel_can(ent_candidata);
    }
8.   object
          {
9.        nombre:=  'OLAP'+model.nombre;
10.        Dim_Sch:= rels ->
                     collect (rs |rs.RelationShip
                     ToDimension (r))->
                     asOrderedSet();
11.        Cub_Sch:= ent_candidatas ->
                     collect(e|e.EntityToCube())->
                     asOrderedSet();
          }
    }
```

Figura A.4. Regla principal

```
   1. mapping EntRel::Entity:: EntityToCube(): Olap::Cube
     {
   2. object
         {
   3.     nombre:='C'+ self.nombre;
   4.     CDA_Cub := self.Rship_Ent->
                     select(rs|rs.multiplicity='*')->
                     collect(rs|rs.RelationShipEndToCDA())->
                     asOrderedSet();
   5.     Mea_Cub := self.Atr_Ent ->
                     select(a|a.tipo='numeric') ->
                     collect (a|a.AttributeToMeasure()) ->
                     asOrderedSet();
              }
      }
```

Figura A.5. Mapping *EntityToCube*

```
1. mapping EntRel::Attribute::AttributeToMeasure():Olap::Measure
   {
2.  object
        {
3.        nombre:= 'M'+self.nombre;
        }
   }
```

Figura A.6. Mapping *AttributeToMeasure*

RelationShipEndToCDA.- El *Mapping RelationShipEndToCDA()* (figura A.7), transforma un rol en una clase de asociación cubo-dimensión (*CubeDimensionAssociation*). El nombre de la clase de asociación, se forma con el nombre del rol al cual se le ha concatenado la cadena de caracteres *´CDA´* (línea 6). En este *Mapping*, se utiliza la instrucción *resolve* (línea 4), para establecer la asociación correcta entre el cubo y la dimensión. La instrucción *resolve*, inspecciona el histórico de la transformación en búsqueda de una instancia de dimensión creada a partir de un rol (*RelationShipEnd)*.

```
1. mapping EntRel::RelationShipEnd:: RelationShipEndToCDA():
   Olap::CubeDimensionAssociation
   {
2.  init {
4.        var rship:= self.resolve(Olap::Dimension)
       }
5.  object
      {
6.       nombre:= 'CDA'+self.nombre;
7.       Dim_CDA:= rship->any(true)
      }
   }
```

Figura A.7. Mapping *RelationShipToCDA*

RelationShipToDim.- El *Mapping RelationShipToDIM()* (figura A.8), transforma una relación (*RelationShip)* en una dimensión.

El nombre de la dimensión, se forma con el nombre de la relación al cual se le ha concatenado el caracter *´D´* (línea 6).

En este *Mapping*, se realizan las siguientes acciones:

1. Se identifican los caminos que pueden generarse a partir de la relación transformada en dimensión. Lo cual, se logra por medio del *Query Genera_Sec_Jer()* (línea 3).

   El resultado de esta función, se asigna a la variable *HierarchyAux*.

2. Se identifican las entidades que serán transformadas en niveles de la dimensión. Lo cual, se logra por medio del *Query Genera_Sec_Level()* (línea 4).

   El resultado de ésta función, se asigna a la variable *LevelAux*.

3. Se transforman las entidades en niveles. Lo cual, se logra por medio del *Mapping EntityToLevel()* (línea 7).

4. Se transforma cada camino identificado en una jerarquía. Lo cual, se logra por medio del *Mapping HierAuxToHierarchy()* (línea 8).

El *Mapping EntityToLevel*(), realiza una iteración sobre los datos almacenados en la variable *LevelAux* y crea un nivel por cada dato.

El *Mapping HierAuxToHierarchy()*, realiza una iteración sobre los datos almacenados en la variable *HierarchyAux* y crea una jerarquía por cada camino almacenado en ella.

```
1. mapping EntRel::RelationShipEnd:: RelationShipToDim
   in rship: EntRel::RelationShipEnd): Olap::Dimension
   {
2.  init {
3.  var HierarchyAux:
          Sequence(Sequence(EntRel::RelationShipEnd)):=
          Genera_Sec_Jer(Sequence{Sequence {self}});
4.  var LevelAux: Set(EntRel::Entity):=
          Genera_Sec_Level(RSENDAux);
     }
5.  object {
6.       nombre:= 'D'+self.Rship_Rsend.nombre;
7.       Lev_Dim:= LevelAux ->
         collect(e|e.EntityToLevel(e)) ->  asOrderedSet();
8.       Hie_Dim:= HierarchyAux->
         collect(h| HierauxToHierarchy(h)) -> asOrderedSet();
      }
   }
```

Figura A.8. Mapping *RelationShipToDIM*

EntityToLevel.- El *Mapping EntityToLevel()* (figura A.9), transforma una entidad en un nivel (línea 1). El nombre del nivel, se forma con el nombre de la entidad al cual se le ha concatenado el carácter 'L' (línea 3).

```
1. mapping EntRel::Entity:: EntityToLevel(): Olap::Level
   {
2.  object {
3.         nombre:='L' + self.nombre
           }
   }
```

Figura A.9. Mapping *EntityToLevel*

HierAuxToHierarchy.- El *Mapping HierAuxToHierarchy()* (figura A.10), transforma una secuencia de elementos tipo *RelationShipEnd* en una jerarquía (línea 1). En este *Mapping,* se realizan las siguientes acciones:

1. Se genera un nombre para la secuencia de elementos de entrada, por medio del *Query Genera_Nombre_Jerarquia()*. Este nombre, es utilizado para formar el nombre de la jerarquía.
2. Se identifican las entidades que se encuentran dentro de la jerarquía y que previamente fueron transformadas en niveles de dimensión por el *Mapping RelationShipToDim()*. Lo cual, se logra por medio del *Query Genera_Sec_Niveles()*. El resultado de este *Query* se asigna a la variable *LevelAux.*

```
1. mapping EntRel::RelationShipEnd::HierAuxToHierarchy
  (in HerAux: Sequence(EntRel::RelationShipEnd)):Olap::Hierarchy
    {
     init {
2.          var nom_jer:String :=
            Genera_Nombre_Jerarquía(HerAux);
3           var LevelAux:Set(EntRel::Entity):=
            Genera_Sec_Niveles(HerAux);
          }
4.    object
          {
5.          nombre := 'H'+nom_jer;
6.          HLA_Hie:= LevelAux->
            collect(rse|rse.RelationShipEndToHieLevAss())->
            asOrderedSet();
          }
    }
```

Figura A.10. Mapping *HierAuxToHierarchy*

3. Se crea una clase de asociación nivel-jerarquía (*HierarchyLevelAssociation*). Lo cual, se realiza por medio del *Mapping RelationShipEndToHieLevAss()* (línea 6).

RelationShipEndToHieLevAss.- El *Mapping RelationShipEndToHieLevAss (),* transforma un rol (*RelationShipEnd)* de una entidad transformada en nivel en una clase de asociación jerarquía-nivel (*HierarchyLevelAssociation*) (figura A.11). El nombre de esta clase, se forma con el nombre del rol al cual se le ha concatenado la cadena de caracteres *´HLA´* (línea 4). En este *Mapping*, se realizan las siguientes acciones:

1. Se establece la asociación correcta entre un nivel y la jerarquía (línea 2). La instrucción *resolve*, inspecciona el histórico de la transformación en búsqueda de una instancia de nivel creada a partir de una entidad.
2. Se identifica el rol que será transformado en una clase *HierarchyLevelAssociation* (línea 3).

```
1.mapping EntRel::Entity::RelationShipEndToHieLevAss (
  in   rship:Set(EntRel::RelationShip)):Olap::HierarchyLevelAssociation
    {
     init
      {
2.       var ent:=  self.resolve(Olap::Level);
3.       var rol:EntRel::RelationShipEnd:= Identifica_Rol( Entity)
      }
     object
      {
4.       nombre:= 'HLA'+rol.nombre;
5.       Lev_HLA := ent->any(true)
      }
   }
```

Figura A.11. Mapping RelationShipEndToHieLevAss

Genera_Sec_Jer.- El objetivo del *Query Genera_Sec_Jer()*, es encontrar todos los posibles caminos a partir del rol de una relación transformada en dimensión (figura A.12).

Esta función, recibe como entrada una secuencia con roles y produce como salida otra secuencia. Cada elemento de la secuencia de salida, representa un camino (como se mencionó anteriormente, cada camino es considerado una jerarquía). La parte principal de esta función, lo constituye el llamado recursivo (línea 10) por medio del cual, la función extiende la búsqueda de roles con multiplicidad igual a muchos a partir del rol que se recibió como entrada. La selección de los roles con multiplicidad igual a muchos, se realiza por medio de la función *Genera_Rutas_Nav()* (figura A.13).

```
1. Query EntRel::RelationShipEnd::Genera_Sec_Jer
   (in jerInicio: Sequence(Sequence(EntRel::RelationShipEnd))):
    Sequence(Sequence(EntRel::RelationShipEnd))
     {
2.     let tamaño_jer: Integer= jerInicio->size() in
3.     let conjunto_jerarquias:
          Sequence(Sequence(EntRel::RelationShipEnd)) =
4.        jerInicio -> iterate(jerarquia_ite;
5.        acc:Sequence(Sequence(EntRel::RelationShipEnd))=
          jerInicio |
6.        acc->union(Genera_Rutas_Nav(jerarquia_ite)))->
         asSequence() in
7.     if conjunto_jerarquias->size() = tamaño_jer then
8.        conjunto_jerarquias;
9.     else
10.       Genera_Sec_Jer(conjunto_jerarquias)
       endif
   }
```

Figura A.12. Query *Genera_Sec_Jer*

```
1. Query EntRel::RelationShipEnd::Genera_Rutas_Nav
   (in rutanav: Sequence(EntRel::RelationShipEnd)):
   Sequence(Sequence(EntRel::RelationShipEnd))
   {
2.  rutanav->last().Rship_Rsend ->
        collect (rsend|rsend.Rsend_Rship)->
        select (ent|ent.Ent_Rship.nombre<>noment)->
        collect (rsend|rsend.Ent_Rship.Rship_Ent)->
        select (rsend|rsend.multiplicity='*')->
3.  iterate(relacion_it;
        acc:Sequence(Sequence(EntRel::RelationShipEnd)) =
        Sequence {} |
        let nueva_tray :Sequence(EntRel::RelationShipEnd)=
        rutanav->append(relacion_it) in
        acc->append(nueva_tray))
   }
```

Figura A.13. Query Genera_Rutas_Nav

Para explicar el funcionamiento de cada *Query*, usaremos el esquema de la figura A.14. A partir del rol *Ext_art_lin* de la relación trasformada en dimensión, la función *Genera_Rutas_Nav()*, identifica dos caminos.

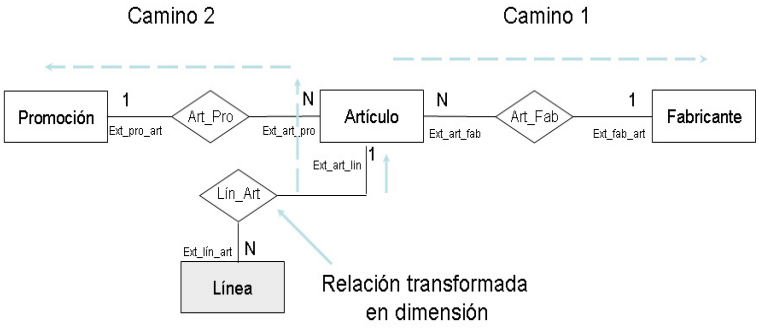


Figura A.14. Caminos en es esquema ER

Cada camino, se compone por una secuencia:

Sequence {Sequence{Ext_art_lin, Ext_art_pro},

Sequence{Ext_art_lin, Ext_art_fab}}

Como se muestra, cada elemento de la secuencia se forma por los roles de la entidad con la que se asocia el rol recibido como parámetro de entrada (en este ejemplo se agregaron los roles de la entidad *Artículo*).

Estos roles, se identifican en la línea 2 de la función *Genera_Rutas_Nav()* y se agregan a la secuencia en la línea 3.

Al finalizar esta función, se devuelve el flujo del programa a la función *Genera_Sec_Jer(),* donde se compara el tamaño de la secuencia de entrada con el tamaño de nueva secuencia (línea 7*)*.

Si la nueva secuencia tiene el mismo tamaño que la secuencia original, se asume que a la secuencia de entrada no le fueron agregados más roles, por lo que finaliza el proceso recursivo.

En caso contrario, se continúa con la iteración sobre los elementos de la secuencia y se continua con la búsqueda de roles con multiplicidad igual a muchos a partir del último elemento de cada secuencia.

En nuestro ejemplo, la búsqueda de roles continuaría a partir del rol *Ext_art_pro* y posteriormente a partir del rol *Ext_art_fab*

IsCandidate.- El objetivo de este Query (figura A.15), es identificar una entidad de cubo candidata. En este *Query*, primero se identifican todas las entidades con las que se asocia la entidad recibida como parámetro de entrada (línea 5).

Si las entidades no tienen roles con multiplicidad igual a muchos (línea 6) y la entidad de entrada si los tiene (línea 8), entonces se regresa un valor de verdad igual a verdadero.

```
1. Query EntRel::Entity::IsCandidate():Boolean
   {
2.    if self.Rship_Ent->size()>0 then
3.      if self.Rship_Ent -> collect (r|r.Rship_Rsend) ->
4.         collect (rs|rs.Rsend_Rship)->
5.         select (rs|rs.nombre <> self.nombre) ->
6.         select (rse|rse.multiplicity='*')->size()<0
7.         self.Rship_Ent->
8.         select (rse|rse.multiplicity='*')->size()>0 then true
4.      else false
        endif
5.    else false
      endif
}
```

Figura A.15. Query *IsCandidate*