Aplicando MDA al Diseño de un Datawarehouse Temporal

Carlos G. Neil
Universidad Abierta Interamericana
Facultad de Tecnología Informática
Buenos Aires, Argentina
carlos.neil@vaneduc.edu.ar

Claudia F. Pons
Universidad Abierta Interamericana
Facultad de Tecnología Informática
CONICET
Buenos Aires, Argentina
cpons@info.unlp.edu.ar

Resumen

Model-Driven Architecture (MDA) es un enfoque ampliamente aceptado para el desarrollo de sistemas de software complejos. MDA propone el uso de modelos en todas las fases de desarrollo, desde la especificación y análisis hasta la implementación. La transformación de modelos es la base de MDA; comenzando por un modelo independiente de la plataforma el objetivo es lograr, en cada paso, modelos más específicos. Adhiriendo a la filosofía MDA, en este artículo, presentamos una metodología para el diseño de un datawarehouse temporal que permite definir los conceptos independientes de la implementación. Nuestro propósito consiste, aplicando el enfoque MDA, en la definición de metamodelos y reglas de transformación formales que provean un marco para el refinamiento de un modelo de datos temporal para la obtención de un esquema relacional.

1. Introduction

Las empresas utilizan los datos acumulados durante años, empleados en las transacciones comerciales, para ayudar a comprender y dirigir sus negocios; con ese propósito, los datos de las diferentes actividades se almacenan y consolidan en una base de datos central denominada datawarehouse; los analistas lo utilizan para extraer información de sus negocios que les permita tomar mejores decisiones [12]. Un datawarehouse es una colección de datos no volátiles, que se acumulan en el tiempo, que están orientados a un tema determinado y que se utilizan para tomar decisiones organizacionales [13]. El modelo multidimensional constituye la base del datawarehouse, en él la información se estructura en hechos y dimensiones; un hecho es un tema de interés para la empresa, se describe mediante atributos denominados atributos de hecho, éstos están contenidos en celdas o puntos en el cubo de datos. Un cubo de datos es una representación multidimensional de datos donde éstos pueden verse desde distintos puntos de vista; está formado por dimensiones que determinan la granularidad para la representación de hechos y jerarquías que muestran cómo las instancias de hechos pueden ser agrupadas y seleccionadas para los procesos de toma de decisión [3].

En el datawarehouse el tiempo es una de las dimensiones para el análisis [8], [9] pero este hace referencia al momento en que se realizó una transacción, no se detalla cuándo, en el mundo real, varían los atributos o interrelaciones involucradas en esas transacciones, La necesidad de registrar valores que permitan evaluar tendencias, variaciones, máximos y mínimos, justifican considerar en el diseño del datawarehouse cómo algunos atributos o interrelaciones pueden variar en el tiempo. Un esquema multidimensional temporal que incluya, además del hecho principal de análisis, esquemas temporales (que no pertenezcan a la jerarquía) permitirá registrar, además, la variaciones temporales de atributos y/o interrelaciones.

Para la construcción del esquema temporal [23] se adaptó un algoritmo que permite en forma semiautomática construir, a partir de un modelo entidad interrelación, el diseño conceptual de un datawarehouse [9]. Se utilizó una extensión del modelo entidad interrelación, ampliándolo con atributos e interrelaciones temporales y, aplicando un algoritmo recursivo, se construyó el esquema conceptual, unificando en un sólo modelo, tanto el esquema multidimensional como el temporal; este esquema permite registrar y analizar las variaciones temporales así como la realización de consultas sobre la estructura multidimensional. La transformación, de

un modelo de datos temporal a un modelo multidimensional temporal, se realizó de manera informal en dos etapas: primero, utilizando el algoritmo recursivo, se creó un grafo de atributos; luego, a partir del grafo de atributos más un conjunto de decisiones de diseño para la determinación de cuáles serán dimensiones, jerarquías y atributos de hecho, se derivó el modelo multidimensional temporal. La realización de estos pasos, si bien están detallados en la metodología de diseño propuesta, no están formalizados de manera conveniente para poder ser automatizados.

Model-Driven Architecture [18] fue establecida como una arquitectura para el desarrollo de aplicaciones; tiene como objetivo proporcionar una solución para los cambios de negocio y de tecnología, permitiendo construir aplicaciones independientes de la implementación; representa un nuevo paradigma en donde se utilizan modelos del sistema, a distinto nivel de abstracción, para guiar todo el proceso de desarrollo. La idea clave subyacente es que, si se trabaja con modelos, se obtendrán importantes beneficios tanto en productividad, portabilidad, interoperatividad y mantenimiento. Podemos dividir el proceso MDA en tres fases; en la primera, se construye un modelo independiente de la plataforma (PIM), este es un modelo de alto nivel del sistema, independiente de cualquier tecnología; luego, se transforma el modelo anterior a uno o más modelos específicos de la plataforma (PSM), este modelo es de más bajo nivel que el PIM y describe al sistema de acuerdo con una tecnología de implementación determinada; por último, se genera el código fuente a partir de cada PSM. La división entre PIM y PSM está vinculado al concepto de plataforma que no está, aún, claramente definido. MDA, además, presenta un modelo independiente de los aspectos computacionales (CIM) que describe al sistema dentro de su ambiente y muestra lo que se espera de él sin exhibir detalles de cómo será construido. El beneficio principal del enfoque MDA es que una vez que se ha desarrollado cada PIM podemos derivar, automáticamente, el resto de los modelos aplicando las correspondientes transformaciones en forma vertical. Sin embargo, pueden aplicarse también transformaciones horizontales; esto es, un modelo fuente se puede transformar en un modelo destino dentro del mismo nivel de abstracción [20]. La transformación de PIM a PIM se utiliza cuando los modelos son ampliados o especializados durante el proceso de desarrollo sin necesidad de información dependiente de la plataforma. Una de las más obvias transformaciones es entre el análisis y el diseño, concepto relacionado con el refinamiento de modelos [18]. Aplicando los conceptos de MDA en la construcción de un datawarehouse, identificamos un CIM que define los requerimientos desde una perspectiva de negocio; un PIM que lo define desde un punto de vista conceptual sin tener en cuenta ningún detalle tecnológico específico y uno o más PSM´s que especifican aspectos de diseño en distintas plataformas, por ejemplo, ROLAP (OLAP Relacional), MOLAP (OLAP Multidimensional) u HOLAP (OLAP Hibrido) [17].

En el presente artículo proponemos, dentro del marco de la filosofía MDA, formalizar la transformaciones presentadas; primeramente, una transformación horizontal (de PIM a PIM), del modelo de datos temporal al modelo multidimensional temporal pasando por un grafo de atributos; luego, realizaremos la transformacion vertical (de PIM a PSM), a una plataforma relacional. Utilizaremos un metamodelo para cada uno de los modelos propuestos y aplicaremos sentencias OCL (Object Constraint Language) [25], para formalizar las transformaciones.

El resto del trabajo está estructurado de la siguiente forma: en el capítulo 2, presentamos la transformación informal del modelo de datos al grafo de atributos, del grafo de atributos al modelo multidimensional y de este último al modelo relacional; en el capítulo 3, mostramos los metamodelos de datos, de grafos, multidimensional y relacional y las tres transformaciones formales; en el capítulo 4, detallamos los trabajos relacionados, tanto los referidos al diseño conceptual de un datawarehouse temporal como a las propuestas de diseño de un datawarehouse en un ambiente MDA; por último, en el capitulo 5, presentamos la conclusión y los trabajos futuros.

2. Transformaciones Informales

La metodología de transformación del modelo de datos temporal al modelo relacional plantea una serie de pasos, descritos de manera informal, que detallaremos a continuación y que, en resumen, consisten en la aplicación de un algoritmo que tiene como entrada un modelo entidad interrelación temporal y, mediante sucesivas transformaciones, obtenemos, primeramente, un modelo multidimensional temporal y, finalmente, un conjunto de tablas relacionales. Presentamos con un ejemplo (Figura 1) cómo, utilizando el algoritmo, transformamos un modelo de datos temporal (Figura 2) a un grafo de atributos (Figura 3); luego, a partir de este, creamos el modelo multidimensional temporal

(Figura 4) y, finalmente, las tablas en el modelo relacional.

Para la aplicación del algoritmo recursivo, primero, transformamos el modelo entidad interrelación (Figura 1) a un modelo entidad interrelación temporal[1] (Figura 2). El atributo multivaluado se convertirá en una entidad débil con una interrelación temporal (marcada con T) y la interrelación temporal se transformará en una entidad con interrelaciones binarias (marcada con T) vinculadas a las entidades participantes [23]. En los casos en que querramos preservar una futura jerarquía, proponemos mantener las dos interrelaciones (la instantánea y la temporal). En el ejemplo (Figura 2), conservamos la interrelación entre *PROVEEDOR* y *LOCALIDAD*. Con el mismo criterio general utilizado para transformar interrelaciones temporales, transformamos la interrelación *venta* en una entidad *VENTA* (Figura 2).

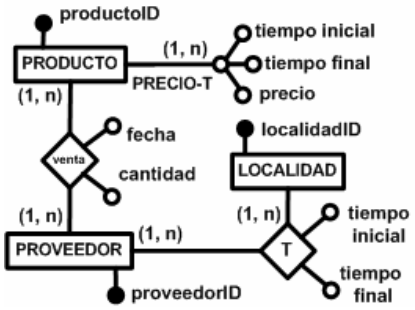

Figura 1. Modelo de Datos

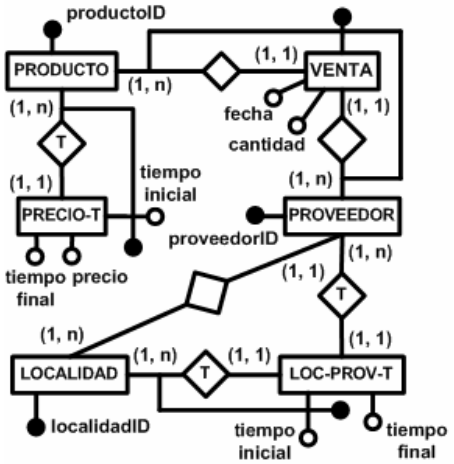

Figura 2. Modelo de Datos Transformado

[1] Por razones de espacio no presentaremos la transformación del modelo de datos al modelo de datos temporal.

2.1 Del Modelo de Datos al Grafo

Los hechos, como conceptos de interés primario para el proceso de la toma de decisión, corresponden a sucesos que ocurren dinámicamente en la realidad, éstos pueden ser representados en el modelo entidad interrelación temporal mediante una entidad E o por medio de una interrelación R n-aria entre entidades E_1... E_n. [9]. Dada un área de interés en un modelo entidad interrelación temporal y una entidad E que pertenece a él, denominamos grafo de atributos al grafo tal que:

- Cada vértice corresponde a un atributo, simple o compuesto del modelo entidad interrelación.
- La raíz corresponde al identificador de E.
- El atributo correspondiente a cada vértice *v*, determina funcionalmente a todos los atributos descendientes de *v*.

Los vértices temporales representan esquemas que tienen como foco de interés la variación de atributos e interrelaciones en función del tiempo. Dado un identifier(E) que indica un conjunto de atributos que identifican a la entidad E, el grafo de atributos (Figura 3) será construido semi automáticamente mediante la aplicación de la siguiente función recursiva modificada de [9]:

```
Function translate (E: Entity): Vertex
{v = newVertex(E);
// newVertex(E) crea un nuevo vértice,
// conteniendo el nombre y el identificador
// del  objeto E
for each attribute a ∈ E| a ∉ identifier(E)
do
   addChild (v, newVertex(a));
   // se agrega un hijo a al vértice v
   for each  entity G connected to E
   by relationship R|card-max(E,R)=1 xor R is
temporal do
// se consideran interrelaciones y atributos
// temporales
{for each  attribute  b ∈ R do
   addChild (v, newVertex(b));
   addChild (v, translate(G));
}
return(v)}
```

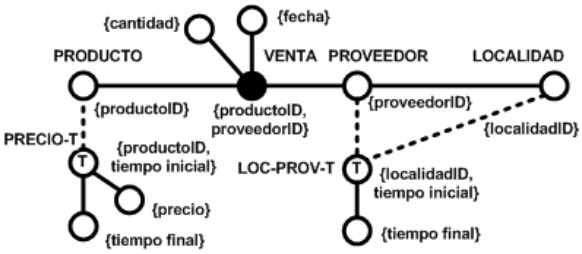

Figura 3. Grafo de Atributos

Cuando ampliamos el modelo entidad interrelación con aspectos temporales, los atributos y las interrelaciones variantes se transformarán en entidades vinculadas con interrelaciones marcadas con T del tipo x-a-muchos (card-max(E, R) > 1); por lo tanto, no podrán ser incluidos en la jerarquía para realizar agregaciones. La línea punteada en el grafo de atributos muestra esta particularidad.

Probablemente no todos los atributos representados en el grafo sean de interés en el datawarehouse. Por tal motivo, este puede ser modificado por el diseñador para eliminar los niveles de detalles innecesarios.

2.2. Del Grafo al Modelo Multidimensional

El proceso de transformación del grafo de atributos al modelo multidimensional temporal, es decir, la elección de cuales vértices del grafo serán atributos de hecho, dimensiones o jerarquías (temporales o no) dependerá de las decisiones del diseñador pero, en general, seguirá el siguiente criterio que utilizaremos en la transformación: la raíz del grafo será el hecho principal; todos los vértices vinculados con la raíz, que no sean identificadores, serán atributos de hecho; los demás atributos vinculados al hecho serán dimensiones; los vértices vinculados a las dimensiones, que no sean identificadores, serán atributos de la dimensión; los demás atributos serán jerarquías (temporales o no) dentro de las dimensiones; todos los atributos vinculados a una jerarquía, si no son identificadores, serán atributos de éstas, sino serán, también, parte de la jerarquía; todas las jerarquías temporales tendrán asociados un rango temporal. El atributo fecha, asociado al hecho, si lo hubiere, será transformado en dimensión. En la figura 4 se muestra el esquema resultante.

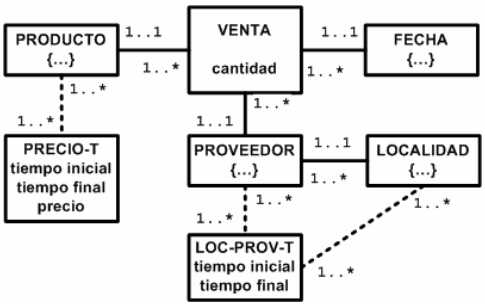

Figura 4. Esquema Multidimensional Temporal

Los atributos e interrelaciones temporales en el grafo (éstos se vinculan mediante líneas punteadas) precisan de una consideración especial en su transformación al esquema de hecho: éstos no formarán parte de la jerarquía para las operaciones de roll-up y drill down, solamente permitirán evaluar cuándo lo atributos e interrelaciones han variado en el tiempo; constituyen, lo que se denomina, jerarquías no estrictas [27].

2.3. Del Modelo Multidimensional al Relacional

Por último, a partir del modelo multidimensional temporal obtendremos, aplicando las siguientes reglas de transformacion, un conjunto de tablas en el modelo relacional. El hecho se transformará en tabla; los atributos de hecho, serán columnas de la tabla; la clave primaria estará compuesta por el conjunto de los atributos identificadores; los atributos que forman la clave primaria serán claves foráneas referenciando a cada una de las tablas resultantes de las transformaciones de las dimensiones del hecho. Las dimensiones se transformán en tablas; los atributos de la dimensiones serán columnas de la tabla; la clave primaria estará compuesta por el conjunto de los atributos identificadores; además, cada tabla dimensión tendrá una clave foránea que hará referencia a cada una de las tablas jerarquía vinculadas a la dimensión. Las jerarquías se transformarán en tablas; los atributos de la jerarquía serán columnas de la tabla; la clave primaria estará compuesta por el conjunto de los atributos identificadores; además, cada tabla jerarquía tendrá una clave foránea que hará referencia a cada una de las jerarquías vinculadas. Las jerarquías temporales se transformarán en tablas. Si la jerarquía temporal deviene de un atributo temporal (isTempAttr = true), tendrá como atributo el tiempo final; la clave primaria será la unión de la clave primaria de la jerarquía vinculada (además, será la clave foránea que hará referencia a dicha tabla jerarquía) más el tiempo inicial. Si la jerarquía temporal deviene de una interrelación temporal (isTempAttr = false), tendrá como atributo el tiempo final y el atributo que es clave primaria de una de las jerarquías vinculada (además, será la clave foránea que hará referencia a la tabla jerarquía); la clave primaria será la unión de la clave primaria de la otra jerarquía vinculada (además, sera la clave foránea que hará referencia a dicha tabla jerarquía) más el tiempo inicial. A continuacion, se presenta el esquema relacional resultante:

VENTA(productoID(PRODUCTO), proveedorID(PROVEEDOR), fechaID(FECHA), cantidad)

FECHA(fechaID,…)
PRODUCTO(productoID, …)
PROVEEDOR(proveedorID, localidadID(LOCALIDAD),…)
LOCALIDAD(localidadID,…)
PRECIO-T(productoID(PRODUCTO), tiempo-inicial, tiempo-final, precio)
LOC-PROV-T(proveedorID(PROVEEDOR), tiempo-inicial, tiempo-final, localidadID(LOCALIDAD)

3. Transformaciones Formales

Una regla de transformación de modelos debe definir, evitando cualquier ambigüedad, la relación implícita que existe entre sus partes. MDA propuso un estándar, QVT (Query, Views, Transformations) [29], que permite crear consultas, vistas y transformaciones de modelos en el marco MDA . Las transformaciones, en el contexto de QVT, se clasifican en *relación* (relation) y *función* (mapping); las relaciones especifican transformaciones multidireccionales, no permiten crear o modificar modelos, pero sí chequear la consistencia entre dos o más modelos relacionados. Las funciones, en cambio, implementan la transformación, es decir, transforma elementos de un dominio en elementos de otro. Se han propuesto varios lenguajes de transformación: BOTL [16]; ATL [14]; Tefkat [15]; Kent Model [1] y también el uso de sentencias OCL para especificar las transformaciones [6], [7]. Todos estos lenguajes asumen que los modelos involucrados en la transformación cuentan con una definición formal de su sintaxis expresada en términos de metamodelos MOF [21]. En este trabajo hemos utilizado el lenguaje declarativo OCL como alternativa a otros lenguajes específicos para expresar la transformación entre los modelos de datos. OCL es suficientemente expresivo y cuenta con una definición más madura y estable que la de los otros lenguajes mencionados.

3.1. Metamodelos

Para la especificación de las reglas de transformación es esencial el conocimiento de los metamodelos, tanto de los modelos fuente como de los modelos destino [18]. UML [30] es ampliamente recomendado y aceptado, aunque no especialmente prescripto, como lenguaje de especificación para modelos MDA.

A continuación, presentaremos los cuatro metamodelos utilizados para las transformaciones: el metamodelo de datos temporal (Figura 5), el metamodelo del grafo de atributos (Figura 6), el metamodelo multidimensional temporal (Figura 7) y el metamodelo relacional (Figura 8). Todas las clases, excepto las del metamodelo del grafo de atributos, heredan el atributo `name` de una superclase `Named`, no mostrada en los graficos.

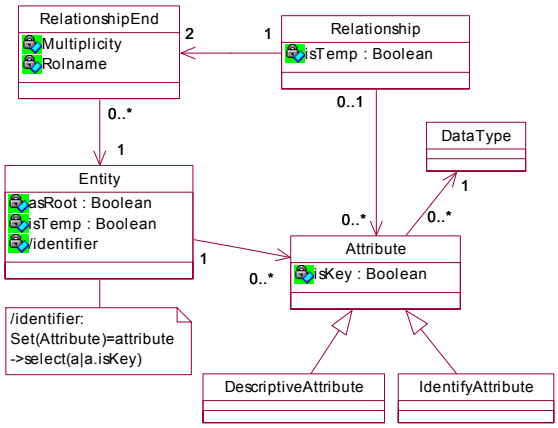

Figura 5. Metamodelo de Datos Temporal

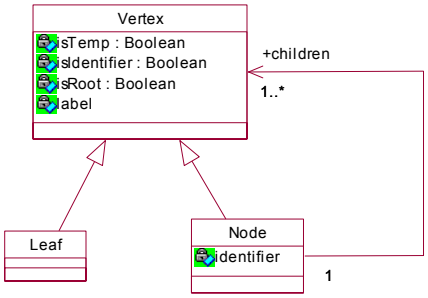

Figura 6. Metamodelo del Grafo de Atributos

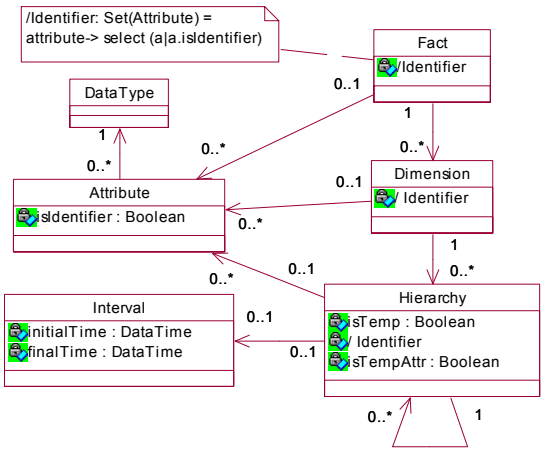

Figura 7. Metamodelo Multidimensional Temporal

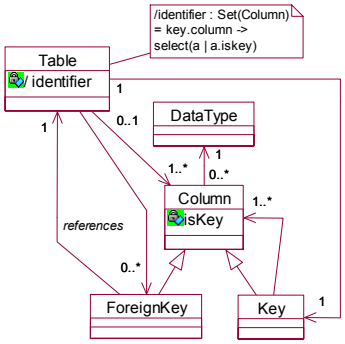

Figura 8. Metamodelo Relacional

3.2. Del Modelo de Datos al Grafo

En esta sección especificaremos formalmente la transformación del modelo de datos al grafo de atributos, la cual fue descrita informalmente en la sección 2.1; utilizaremos los metamodelos de las figuras 5 y 6 y el lenguaje OCL para definir una funcion toVertex() que, al ser aplicada sobre un objeto de tipo Entity perteneciente al modelo de datos temporal, retorna un objeto de tipo Vertex correspondiente al grafo de atributos. La función toVertex() se especifica mediante la siguiente construcción OCL:

```
Context e:Entity :: toVertex(): Vertex
post:   e.name = result.label and
e.identifier = result.identifier
-- los atributos de e se convierten
-- en hijos de result
and e.attribute -> forall(a | e.identifier ->
includes(a) or   result.children ->
includes(toLeaf())
-- Se consideran atributos e
-- interrelaciones temporales
and e.connections -> forAll( Tuple{r, g} |
(card-max(e, r) = 1 xor r.isTemp) implies
(r.attribute -> forall(b | result.children ->
includes(toLeaf()))
and result.children -> includes(g.toVertex())
)
```

Definiciones adicionales al metamodelo:

```
Context Entity :: identifier: Set(Attribute)
-- retorna el conjunto de atributos
-- identificadores de la entidad;

Context Entity :: connections: Set(TupleType
(r: Relationship, g: Entity))
-- retorna el conjunto de todas las posibles
-- tuplas {r, g} donde r es una interrelación
-- vinculada a self, en tanto que g sea una
-- entidad conectada al extremo opuesto de r

Context a: Attribute :: toleaf() : Leaf
result.name = a.name
-- transforma el atributo en hoja del grafo
```

3.3. Del Grafo al Modelo Multidimensional

La transformación del grafo de atributos al modelo multidimensional, tal como fue detallada en la sección 2.2, será formalmente especificada utilizando expresiones OCL sobre los metamodelos de las figuras 6 y 7, de la siguiente forma:

```
Context v: Vertex :: toFact(): Fact
Pre: v.isRoot
-- la raíz del grafo se transforma en  el
-- hecho principal
Post: result.name = v.label
-- los vértices vinculados con la raíz, que
-- no sean identificadores, se transformarán
-- en las   medidas del hecho; los demás
-- atributos vinculados al hecho, serán
-- dimensiones; el atributo fecha, asociado
-- al hecho (si lo hubiere), será
-- transformado en dimension
and v.children -> forAll( w | if
(w.isIdentifier or w.label = "fecha")
then
result.dimension -> includes(w.toDimension())
else
result.attribute -> includes(w.toAttribute())
endif)

Context v: Vertex :: toAttribute(): Attribute
Post: result.name = v.label

Context v: Vertex :: toDimension(): Dimension
Post: result.name = v.label
-- los vértices vinculados a las dimensiones
-- que no sean identificadores,serán tributos
-- de la dimensión; los demás atributos serán
-- jerarquías dentro de las dimensiones
and v.children -> forAll( w | if
w.isIdentifier  then
result.hierarchy -> includes(w.toHierarchy())
else
result.attribute -> includes(w.toAttribute())
endif)

Context v: Vertex :: toHierarchy(): Hierarchy
Post: result.name = v.label
-- las jerarquías temporales tendrán
-- asociados un rango temporal
and v.isTemp implies result.isTemp and
result.interval -> notEmpty()
-- los atributos vinculados a una jerarquía,
-- si no son identificadores, serán atributos
-- de éstas, sino, serán parte  de la
-- jerarquía;
and v.children -> forAll(w | if
w.isIdentifier then
result.hierarchy -> includes(w.toHierarchy())
else
result.attribute -> includes(w.toAttribute())
endif)
```

3.4. Del Modelo Multidimensional al Relacional

La transformación del modelo multidimensional al relacional, tal como fue detallada en la sección 2.3, será formalmente especificada utilizando expresiones

OCL sobre los metamodelos de las figuras 7 y 8, de la siguiente forma:

```
Context f: Fact :: toTable(): Table
Post:
-- el hecho se transforma en tabla
result.name = f.identifier
-- los atributos de hecho serán
-- columnas de la tabla
result.column = f.attribute ->
collect(a | a.toColumn())
-- la clave primaria estará compuesta por el
-- conjunto de los atributos identificadores
result.key = f.attribute ->
select(a | a.isIdentifier) ->
collect (i | i.toColumn())
-- los atributos que forman la clave primaria
-- serán, además, claves foráneas
-- referenciando a cada una de las tablas
-- resultantes de las transformaciones de las
-- dimensiones del hecho.
result.foreignKey = result.key and
result.foreignKey -> collect(k | k.table) =
f.dimension -> collect(d | d.toTable())

Context d:Dimension :: toTable(): Table
Post:
-- Las dimensiones se transformarán en tablas
result.name = d.identifier
-- los atributos de la dimensiones serán las
-- columnas de la tabla
result.column = d.attribute ->
collect(a | a.toColumn())
-- la clave primaria estará compuesta por el
-- conjunto de los atributos identificadores
result.key = d.attribute ->
select(a | a.isIdentifier) ->
collect (i | i.toColumn())
-- cada tabla dimensión tendrá claves
-- foráneas que harán referencia a cada una
-- de las jerarquías vinculadas a la
-- dimensión
result.foreignKey -> collect(k | k.table) =
d.hierarchy -> collect(h | h.toTable())


Context h:Hierarchy :: toTable(): Table
Post:
-- las jerarquías se transforman en tablas
result.name = h.identifier
if h.isTemp = true
then
-- Caso 1: transformacion de jerarquias no
-- temporales: los atributos de la jerarquía
-- serán las columnas de la tabla
result.column = h.attribute ->
collect(a | a.toColumn())
-- la clave primaria estará compuesta por el
-- conjunto de los atributos identificadores
result.key = h.attribute ->
select(a | a.isIdentifier) ->
collect (i | i.toColumn())
-- cada tabla jerarquía tendrá una clave
-- foránea que hará referencia a cada una de
-- las jerarquías vinculadas
result.foreignKey -> collect(k | k.table) =
h.hierarchy -> collect(h | h.toTable())
else
if (isTempAttr = true)
then
-- Caso 2: transformación de jerarquía
-- temporal que proviene de un atributo
-- temporal: tendrá como columna al tiempo
-- final;
result.column = h.attribute ->
union Set{h.interval.finalTime} ->
collect(a | a.toColumn())
-- la clave primaria será la unión de  la
-- clave primaria de la jerarquía más el
-- tiempo inicial
result.key = h.attribute ->
select(a | a.isIdentifier) -> union
Set{h.interval.initialTime}->
collect (i | i.toColumn())
else
-- Caso 3: transformación de jerarquía
-- temporal que deviene de una interrelación
-- temporal: tendrá como columna al tiempo
-- final y al atributo que es clave primaria
-- de la jerarquía vinculada; la clave
-- primaria será la unión de la  clave
-- primaria de la otra jerarquía  vinculada
-- más el tiempo inicial.
endif
endif

context a: Attribute :: toColumn(): Column
result.name = a.name
-- transforma el atributo en columna de la
-- tabla
```

4. Trabajos Relacionados

Se propusieron varias soluciones considerando los aspectos temporales en el Datawarehouse; en [5] se presentó un esquema estrella temporal que difiere del tradicional en cuanto al tratamiento del tiempo, mientras este toma al tiempo como una dimensión más, aquel anula la dimensión tiempo y agrega, como atributos de hecho, el tiempo inicial y el final en cada una de las filas de las tablas del esquema. En [27] se describió, entre las características que un modelo de datawarehouse debería tener, la necesidad de considerar los cambios temporales en los datos y las jerarquías no estrictas. En [19] se presentó el modelo multidimensional temporal y un lenguaje de consulta temporal, donde se agregan marcas de tiempo en las dimensiones o al nivel de instancias (o ambos) para capturar las variaciones en los atributos de las dimensiones.

Entre los trabajos vinculados a la transformación de modelos, en [11] se describió, mediante Meta Object Facility (MOF), la transformación del esquema entidad interrelación al esquema relacional utilizando sentencias OCL para establecer restricciones en el metamodelo. En [4] se plantearon dos fases para la migración de un sistema relacional a un sistema de base de datos orientado a objetos; en la primera, utiliza reglas de transformación para construir un esquema OO que es semánticamente equivalente al esquema relacional, en la segunda fase ese esquema es

usado para generar programas que migren los datos relacionales a una base de datos orientado a objetos. En [10] se estudió la sintaxis y la semántica del modelo entidad interrelación y el modelo de datos relacional y sus transformaciones. En [26] se mostró un marco para representar metadatos acerca de datos fuentes, datos destinos, transformaciones y los procesos y operaciones que crean y administran un datawarehouse. En [2] se estudió el problema de la traducción de esquemas entre diferentes modelos de datos, introducen un formalismo teórico gráfico que permite representar uniformemente esquemas y modelos para comparar diferentes modelos de datos y describir el comportamiento de la traducción. En [23] se estableció una conexión formal entre modelos de datos; se utilizaron técnicas de metamodelo basado en MOF para representar la transformación, mediante un algoritmo, del esquema entidad interrelación temporal al modelo multidimensional temporal; se emplearon diagramas de clases MOF y sus correspondientes reglas OCL para establecer restricciones en el modelo y en el metamodelo. En [28] se definió una estrategia para verificar formalmente la corrección de transformaciones entre modelos en el contexto de MDE.

Existen trabajos donde, específicamente, se utilizó el enfoque MDA para el diseño de un datawarehouse. En [17] se presentó un método estándar e integrado para el diseño de un datawarehouse; se definió el MMD^2A (MultiDimensional Model Driven Architecture) como un enfoque para la aplicación del marco MDA en el modelado multidimensional. En [29] se propuso un método para el diseño conceptual de un datawarehouse, planteado en tres fases: en la primera se extraen un conjunto de esquemas multidimensionales de las bases de datos operacionales mediante reglas de transformaciones definidas en el marco de MDA, la segunda fase está vinculada con la identificación y la elección de los requisitos del usuario; por último, estos requisitos se usan para seleccionar y refinar los esquemas multidimensionales. En [24] se presentó, utilizando metamodelos, reglas de transformacion y aplicando el enfoque MDA, una metodología que convierte un modelo entidad interrelacion temporal en un esquema multidimensioneal temporal.

5. Conclusión y Trabajos Futuros

MDA promueve el uso intensivo de modelos en el proceso de desarrollo, se construyen modelos de los sistemas utilizando primitivas de alto nivel de abstracción, luego estos modelos son transformados hasta obtener código fuente del sistema final. Inicialmente, se crea un modelo independiente de la plataforma (PIM); luego, se transforma el modelo anterior a uno o más modelos específicos de la plataforma (PSM) y, por último, se genera el código a partir de cada PSM. En el presente trabajo se desarrolló una metodología semiautomática para generar un esquema relacional de un datawarehouse temporal (ROLAP) a partir de un modelo de datos temporal; primero se presentó un algoritmo recursivo que permitió armar un grafo de atributos a partir de un modelo de datos; luego, se estableció informalmente la transformación del árbol de atributos al modelo multidimensional y de este al esquema relacional; a continuación, se presentaron los metamodelos del modelo de datos temporales, del grafo de atributos del modelo multidimensional y del relacional. Finalmente, utilizando sentencias OCL, se detallaron las transformaciones formales.

En trabajos futuros se desarrollarán reglas de transformación (mapping) que permitan implementar las relaciones (relation) presentadas en este trabajo y en el desarrollo de plug-ins en la plataforma Eclipse que permita implementar el esquema relacional en diferentes DBMS.

Referencias

[1] Akehurst D.H., Howells W.G.J., McDonald-Maier K.D. Kent Model Transformation Language Proc. Model Transformations in Practice Workshop, part of MoDELS 2005, Montego Bay, Jamaica. 2005

[2] Atzeni P, Torlone R., Schema Translation Between Heterogeneous Data Models in a Lattice Framework. 6th IFIP TC-2 Working Conference on Database Semantics (DS-6), Atlanta, Georgia, 1995

[3] Agrawal R, Gupta A, Sarawagi S., Modeling Multidimensional Databases, Research Report, IBM Almaden Research Center, San Jose, California, 1995.

[4] Behm, A., Geppert, A., Dittrich, K. R. "On the Migration of Relational Schemes and Data Object-Oriented Database System". In Proceedings of Re-Technologies in Information System. Klagenfurt, Austria, Dec 1997.

[5] Bliujute R., Saltenis S., Slivinskas G., and Jensen C. S., Systematic Change Management in Dimensional Data Warehousing. in Proceedings of the Third International Baltic Workshop on Data Bases and Information Systems, Riga, Latvia, 1998.

[6] Cariou, E., Marvie, R., Seinturier, L., & Duchien, L. (2004). OCL for the Specification of Model Transformation Contracts. In J. Bezivin (Eds.), Proceedings of OCL&MDE'2004, OCL and Model Driven Engineering Workshop. Lisbon, Portugal. 2004.

[7] Cariou, E., Marvie, R., Seinturier, L., & Duchien. Model Tranformation Contracts and their Definition in UML and OCL. Technical Report 2004-08, 2004.

[8] Chaudhuri S. and Dayal U., An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record 26(1), March 1997.

[9] Golfarelli M., Maio D., Rizzi S., The Dimensional Fact Model: a Conceptual Model for Data Warehouses. International Journal of Cooperative Information Systems, vol 7, n.2&3, 1998.

[10] Gogolla Martin, Lindow Arne, Richters Mark, Ziemann Paul: Metamodel Transformation of Data Models, Workshop in Software Model Engineering, 2002.

[11] Gogolla Martin, Lindow Arne: Transforming Data Models with UML, IOS Press, 2003.

[12] Gupta, H., Harinarayan, V. Rajaraman, A. and J. Ullman. Index Selection for OLAP. Proceeding ICDE 1997.

[13] W. Inmon. Building the Data Warehouse. John Wiley & Sons, 2002.

[14] Jouault, F, Kurtev, I: Transforming Models with ATL. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica.

[15] Lawley Michael, Steel Jim. Practical Declarative Model Transformation with Tefkat, Lecture Notes in Computer Science, Volume 3844, Jan 2006

[16] Marschall Frank, Braun Meter: Model Transformations for the MDA with BOTL In: Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications, CTIT Technical Report TR-CTIT-03-27, Univeristy of Twente, June 2003.

[17] Mazón Jose Norberto, Trujillo Juan, Serrano Manuel, Piattini Mario: Applying MDA to the Development of Data Warehouses. DOLAP 2005: 57-66.

[18] MDA. Model Driven Architecture. 2004. http://www.omg.org/cgi-bin/doc/formal/03-06-01.

[19] Mendelzon A, Vaisman. A Temporal Query in OLAP. VLDB 2000: 242-253.

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

[21] MOF. Mata Object Facility 1.3. OMG (1999)

[22] Neil Carlos, Ale Juan. A Conceptual Design for Temporal Data Warehouse. 31º JAIIO. Santa Fe. Simposio Argentino de Ingeniería de Software. 2002.

[23] Neil Carlos, Pons Claudia. Formalizing the Model Transformation Using Metamodeling Techniques ASSE Argentinean Symposium on Software Engineering. (33 JAIIO04) September 2004. Cordoba. Argentina.

[24] Neil Carlos, Pons Claudia. Diseño Conceptual de un Datawarehouse Temporal en el Contexto de MDA. XII Congreso Argentino de Ciencias de la Computación. CACIC. San Luis. Argentina. 2006

[25] OCL. Object Constraint Language - version 2.0. 2003.

[26] OMG, ed: The Common Warehouse Metamodel Especification. OMG (2000). www.omg.org.

[27] Pedersen T. B., Jensen C. S, Multidimensional Data Modeling for Complex Data. 1998. ICDE 1999

[28] Pons C. and Garcia D. "An OCL-based Technique for Specifying and Verifying Refinement-oriented Transformations in MDE". Proceedings MoDELS/UML 2006 "Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genoa, Italy, October 2006" LNCS.

[29] OMG. Meta object facility (MOF) 2.0 Query/View/Transformation Specification. OMG Document ptc/05-11-01, Nov 2005.

[30] UML. The Unified Modeling Language Specification – version 2.0. 2003.

[31] Zepeda Leopoldo, Celma Matilde: Aplicando MDA al Diseño Conceptual de Almacenes de Datos. JIISIC 2006: 271-278