Usando ATL en la Transformación de Modelos Multidimensionales Temporales

Carlos Neil [1]
carlos.neil@vaneduc.edu.ar

Martin Baez [2]
mbaez@lifia.info.unlp.edu.ar

Claudia Pons [1]
cpons@info.unlp.edu.ar

[1] Facultad de Tecnología Informática
Universidad Abierta Interamericana
Buenos Aires, Argentina

[2] Facultad de Informática
Universidad Nacional de La Plata
Buenos Aires, Argentina

Abstract

Model-Driven Architecture (MDA) is a widely accepted approach to the complex software systems design. MDA proposes the use of models in every software development phase, from specification and analysis to implementation. Model transformation is the foundation of MDA, starting from a platform independent model, the aim is to achieve more specific models, in each step. According to this philosophy we present a temporal multidimensional design methodology which allows us to define concepts independently of any implementation issue. In the present work our aim is using ATL (Atlas Trasformation Language) to define both the metamodel and the transformation rules for providing a framework to derive a relational logical schema from an abstract temporal data model. Additionally we are designing an Eclipse plug-in for implementing the transformation.

Keyworks: Temporal Multidimensional Model, Model Transformation, MDA, ATL.

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 esta filosofía, presentamos una metodología para el diseño de un datawarehouse temporal que permite definir los conceptos independientemente de la implementación. En el presente trabajo, nuestro propósito consiste en la definición de metamodelos y reglas de transformación usando ATL (Atlas Trasformation Language) que provean un marco para la derivación de un esquema lógico relacional a partir de un modelo de datos conceptual temporal. Además, estamos diseñando un plug-in en Eclipse para implementar dicha transformación.

Palabras claves: Modelo Multidimensional Temporal, Transformación de Modelos, MDA, ATL.

1 INTRODUCCIÓN

Model-Driven Architecture [17] se ha establecido como una arquitectura para el desarrollo de sistemas informáticos; tiene como objetivo brindar una solución para los cambios, tanto de negocio como de tecnología, permitiendo construir aplicaciones independientes de su posterior implementación; representa un nuevo paradigma en donde se utilizan modelos del sistema, en distintos niveles de abstracción, para guiar todo el proceso de desarrollo. Un tipo particular de aplicación, denominado sistemas OLAP (On-Line Analytical Processing), que analiza datos consolidados en un almacenemiento, denominado datawarehoue, es particularmente apropiado para desarrollarse con este enfoque. Los sistemas OLAP se caracterizan por la realización de consultas sobre estructuras de almacenamiento y, aunque en general, estas estructuras de datos pueden ser complejas, la definicion de sus modelos son relativamente simples y, por lo tanto, también lo será la formalización de la definición de las transformaciones utilizadas en MDA.

La idea clave subyacente en MDA 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), éste es un modelo del sistema de alto nivel, independiente de cualquier tecnología; luego, se transforma el modelo anterior a uno o más modelos específicos de la plataforma (PSM), éste 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. 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 transformará en un modelo destino dentro del mismo nivel de abstracción [19]. La transformación de PIM a PIM se utiliza cuando los modelos son ampliados o especializados durante el proceso de desarrollo, sin necesidad de contar con información dependiente de la plataforma. Una de las más obvias transformaciones es la que se realiza entre el análisis y el diseño, concepto relacionado con el refinamiento de modelos [17].

El datawarehouse es una colección de datos no volátiles, que varían en el tiempo, que están orientados a un tema determinado y que se utiliza para tomar decisiones organizacionales [8]. 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 éste hace referencia al momento en que se realizó una transacción, no se detalla cómo ni cuándo 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. Por lo tanto, 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 [22] 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 tres 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. Por último, se establecieron criterios para derivar un modelo logico relacional.
Aplicando los conceptos de MDA en la construcción de un datawarehouse, identificamos un CIM que especifica 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 detallan aspectos de diseño en distintas plataformas, por ejemplo, ROLAP (OLAP Relacional), MOLAP (OLAP Multidimensional) u HOLAP (OLAP Hibrido) [16].
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, una transformacion vertical (de PIM a PSM), a una plataforma relacional. Utilizaremos un metamodelo para cada uno de los modelos propuestos y aplicaremos el lenguaje ATL [12], 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 dos 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 DESCRITAS INFORMALMENTE

La metodología de transformación del modelo de datos temporal al modelo relacional plantea una serie de pasos, descritos originalmente de manera informal [21], 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) en un grafo de atributos (Figura 3); luego, a partir de éste, 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) vinculada a la entidad que poseía el atributo temporal; la interrelación temporal se transformará en una entidad con interrelaciones binarias (marcada con T) vinculadas a las entidades participantes [22]. En los casos en que queramos preservar una futura jerarquía, proponemos mantener las dos interrelaciones

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

(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).

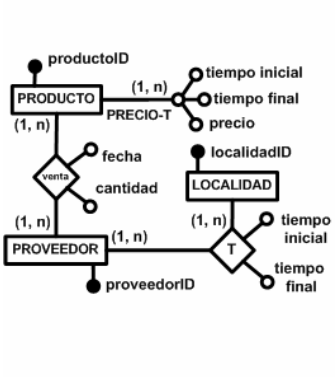

Figura 1. Modelo de Datos

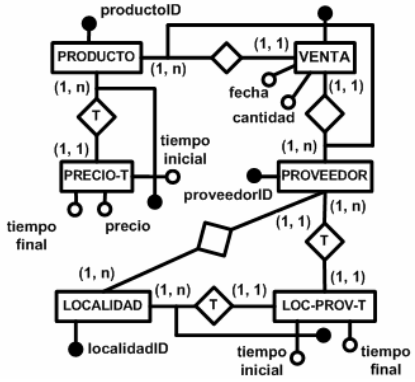

Figura 2. Modelo de Datos Transformado

2.1 Transformación del Modelo de Datos al Grafo de Atributos

Los hechos, como conceptos de interés primario en el proceso de 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);
}
```

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; por lo tanto, no podrán ser incluidos en la jerarquía para realizar operaciones de agregación. La línea punteada en el grafo de atributos muestra esta particularidad. Por último, probablemente, no todos los atributos representados en el grafo sean de interés para el diseño del datawarehouse. Por tal motivo, éste puede ser modificado para eliminar los niveles de detalles innecesarios [9].

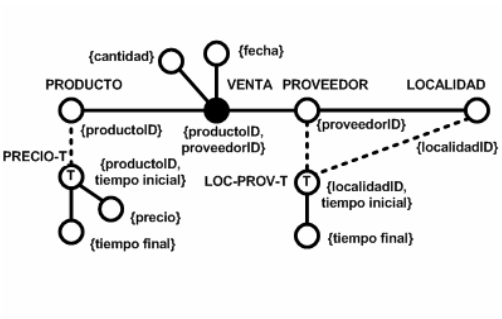

Figura 3. Grafo de Atributos

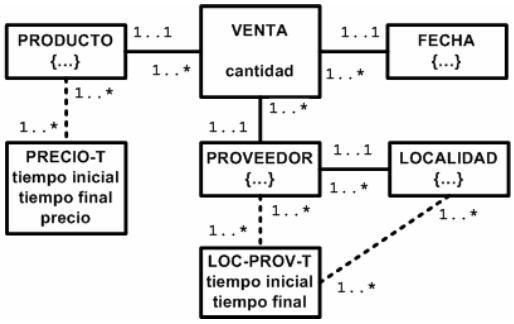

Figura 4. Esquema Multidimensional Temporal

2.2 Transformación del Grafo de Atributos al Modelo Multidimensional

El proceso de transformación del grafo de atributos al modelo multidimensional temporal, esto es, la elección de cuáles 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. 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 cómo atributos e interrelaciones han variado en el tiempo; constituyen, lo que se denomina, jerarquías no estrictas [26].

2.3 Transformación del Modelo Multidimensional al Modelo 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, además, 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 jerarquías 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 tablas 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 tabla 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 tablas jerarquías vinculadas (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 tabla 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 TRASFORMACIONES DESCRITAS FORMALMENTE

Una regla de transformación de modelos debe definir, evitando cualquier ambigüedad, la relación implícita que existe entre sus partes. MDA no especifica ni prescribe ningún lenguaje para la transformación de modelos. El estándar actualmente establecido por OMG para crear consultas, vistas y transformaciones de modelos es QVT (*Query, Views, Transformations*) [28]. 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 [15]; ATL [12]; Tefkat [14]; Kent Model [1] y también el uso de sentencias OCL [25] 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 [20]. En este trabajo hemos utilizado ATL (*Atlas Trasformation Language*) para describir la transformación formal entre los modelos de datos.

3.1 El Lenguaje de Transformación Atlas (ATL)

La transformación de modelos es un componente crítico en MDA; OMG así lo ha reconocido y como respuesta, a través de QVT RFP, han aparecido diversas propuestas. El objetivo de este llamado era definir un lenguaje capaz de expresar consultas, vistas y transformaciones sobre modelos en el contexto de la arquitectura del metamodelado MOF 2.0. La sintaxis abstracta de QVT está definida como un metamodelo MOF; este metamodelo define tres sublenguajes que, en conjunto, forman un lenguaje de transformación híbrido con construcciones imperativas y declarativas denominados: *Relations, Core* y *Operacional Mapping*. Los dos primeros son declarativos, con dos niveles diferentes de abstracción. La documentación de la especificación define la sintaxis textual concreta y la sintaxis abtracta. Además, el lenguaje *Relations* tiene una sintaxis gráfica. El *Operacional Mapping* es un lenguaje imperativo que extiende a los lenguajes *Relations* y *Core.* [13]. ATL comparte las caracteristicas comunes y el mismo conjunto de

requerimientos definidos en QVT RFP. ATL es un framework para administrar transformaciones basadas en modelos. Es un lenguaje mixto, es decir, es una mezcla de construcciones imperativas y declarativas. Un modelo fuente se transforma en un modelo destino mediante una definición de transformación escrita en ATL, que también es un modelo. Los modelos fuente, destino y la definición de la transformación, responden a sus metamodelos respectivos y, a su vez, todos los metamodelos se ajustan a MOF. La transformación de ATL es unidireccional, opera sobre un modelo fuente de sólo lectura y produce un modelo destino de sólo escritura. Durante la ejecución de una transformación, el modelo fuente puede ser navegado pero no cambiado, en cambio el modelo destino no puede ser navegado. Una transformación bidireccional puede implementarse como un par de transformaciones, una para cada dirección [12]. La arquitectura de ATL está compuesta por tres capas; son descritas en niveles decrecientes de complejidad como: *Atlas Model Weaving* (AMW), ATL y ATL *Virtual Machine* (ATL VM). Los programas ATL compilados son ejecutados mediante la ATL VM, que utiliza un conjunto de instrucciones orientadas al modelo. AMW puede ser, opcionalmente, utilizado como un lenguaje de especificación de transformaciones de alto nivel de abtracción [13].
Eclipse [29] es una herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrolo (IDE). ATL ha implementado un ambiente de desarrollado sobre la plataforma Eclipse; las herramientas disponibles para la transformación de modelos son principalmente dos: el núcleo de funcionalidades ATL, que incluye el motor de transformación y las facilidades de admnistración de modelos. La parte básica de ATL incluye todos los componentes requeridos para configurar y ejecutar transformaciones, en particular, el EMF (*Eclipse Modelling framework*) y MDR (*Meta Data repository*) que permiten, respectivamente, manejar modelos definidos de acuerdo acuerdo a la semántica Ecore y MOF 1.4. Las herramientas basicas de ATL también incluyen una notacion textual simple y la notación *Kernel MetaMetaModel* (KM3) que permite el diseño de metamodelos en forma textual. Las facilidades de manejo de modelos está provista por *Atlas MegaModel Management* (AM3), este módulo tiene como objetivo la administración de los recursos globales en el ambiente de la ingenieria conducida por modelos.

3.2 Metamodelos Usados en la Transformación

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 [17]. 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.
A modo de ejemplo, describiremos el metamodelo de datos temporal (figura 5) utilizando KM3; además, ejecutado en la plataforma Eclipse, mostramos la generación del metamodelo en Ecore y su diagrama UML (figura 9).

```
package MMDatosTemporal {
class Relationship {
        attribute isTemp : Boolean;
        attribute name : String;
        reference attributes[0-*] ordered container : Attribute ;
       reference relationshipEnds[2-2] ordered container: RelationshipEnd oppositeOf relationship;
}

class RelationshipEnd {
        attribute multiplicity : Integer;
        attribute rolName : String;
```

```
        attribute name : String;
        reference relationship : Relationship oppositeOf relationshipEnds;
        reference entity : Entity ;
}

class Entity {
        attribute name : String;
        attribute asRoot : Boolean;
        attribute isTemp : Boolean;
        reference attributes[0-*] ordered container : Attribute ;
}

abstract class Attribute {
        attribute name : String;
        attribute isKey : Boolean;
        reference types[0-1] : DataType ;
}

class DescriptiveAttribute extends Attribute {}

class IdentityAttribute extends Attribute {}

class DataType {
       attribute name : String;
}
}
```

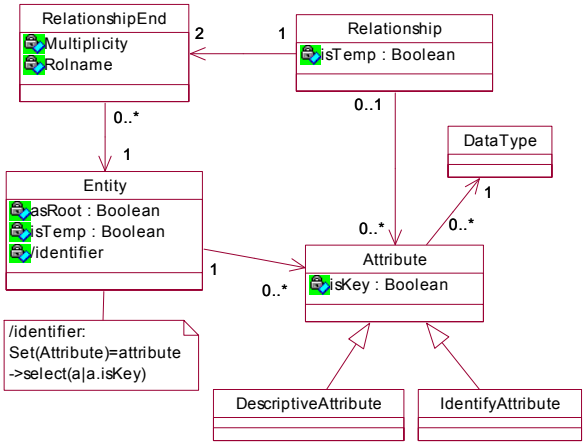

Figura 5. Metamodelo de Datos Temporal

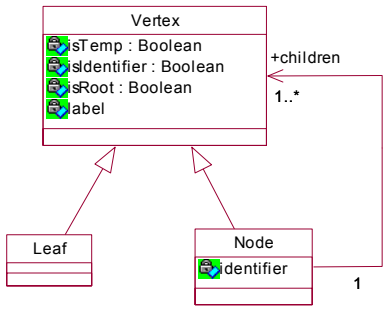

Figura 6. Metamodelo del Grafo de Atributos

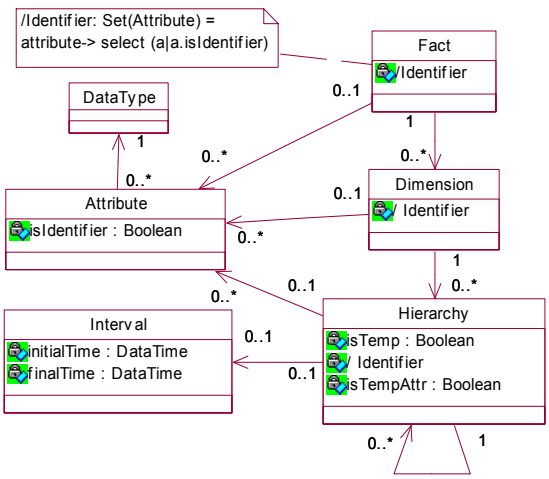

Figura 7. Metamodelo Multidimensional Temporal

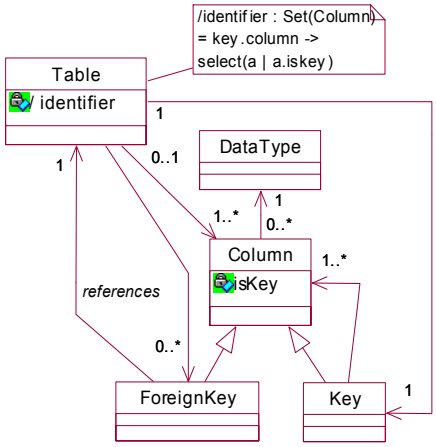

Figura 8. Metamodelo Relacional

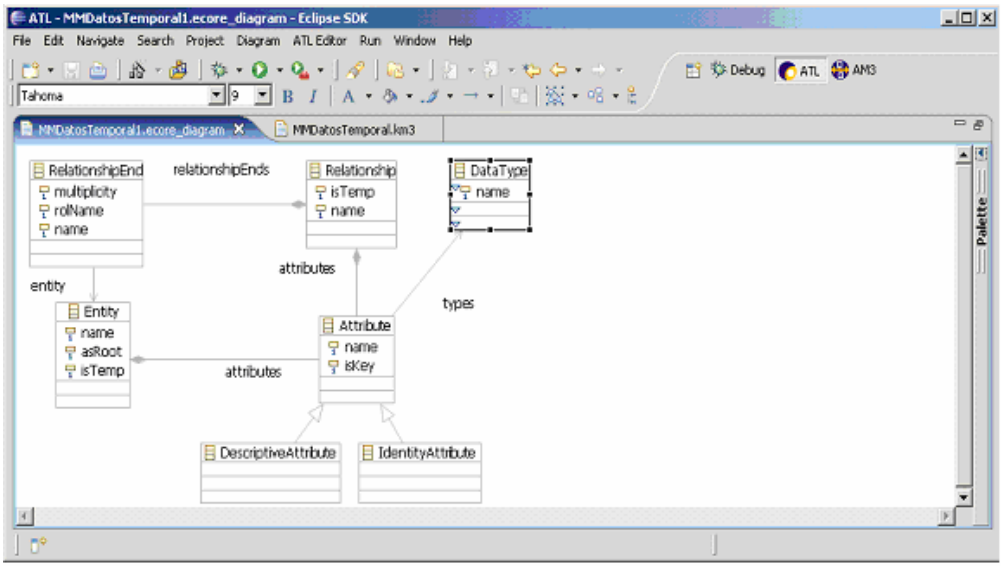

Figura 9. Metamodelo Generado en Eclipse

3.3 Transformaciones Usando ATL

En esta sección, especificaremos formalmente dos transformaciones: la primera, del modelo de datos al grafo de atributos, que ya fue descrita informalmente en la sección 2.1. La segunda, del grafo de atributos al modelo multidimensional, tal como fue detallada en la sección 2.2.

3.3.1 Transformación del Modelo de Datos al Grafo de Atributos

```
module MDatoSTemporales2MGrafo; -- Module Template
create OUT : MMGrafo from IN : MMDatosTemporal;
-- SECCIÓN DE HELPERS ----------------------------------------------------------------------
-- retorna una colección de tuplas(r,g) donde r es una interrelación vinculada a self
-- y g es la entidad del extremo opuesto, con cardinalidad no mayor a 1 o r es temporal

helper context MMDatosTemporal!Entity def: connections():
       Set(TupleType(r : MMDatosTemporal!Relationship, e: MMDatosTemporal!Entity)) =
       self.relationshipEnds
-- Se filtran las relaciones cuya multiplicidad es 1 o son temporales
       -> select (r | r.multiplicity=1 xor r.relationship.isTemp) ->collect(r | r.toTuple());
-- El resultado obtenido se convierte en tupla

helper context MMDatosTemporal!Entity def: identityAttributes():
       Set(MMDatosTemporal!IdentityAttribute) =
       self.attributes -> select (r | r.oclIsTypeOf(MMDatosTemporal!IdentityAttribute));
-- Se filtran las relaciones cuya multiplicidad es 1 o son temporales

helper context MMDatosTemporal!Entity def: descriptiveAttributes():
       Set(MMDatosTemporal!DescriptiveAttribute) =  self.attributes
-- Se filtran las relaciones cuya multiplicidad es 1 o son temporales
       -> select (r | r.oclIsTypeOf(MMDatosTemporal!DescriptiveAttribute));

-- Convierte una RelationshipEnd en una tupla {r,g} donde r es  la relacion y g la entidad
-- del extremo opuesto.
helper context MMDatosTemporal!RelationshipEnd def: toTuple():
       TupleType(r : MMDatosTemporal!Relationship, e: MMDatosTemporal!Entity)=
       Tuple{r=self.relationship,e=self.relationship.entity};

-- Dada una tuple {r,g} donde r es  una relación y g la entidad
-- del extremo opuesto retorna g.
helper context TupleType(r : MMDatosTemporal!Relationship, e: MMDatosTemporal!Entity)
def: getEntity(): MMDatosTemporal!Entity = self.e;

-- Regla called,dado un nombre crea un Nodo hoja del grafo(Leaf).
rule  Descriptive2Leaf(nombre:String) {
to    aLeaf : MMGrafo!Leaf
```

```
do {
        aLeaf.label <- nombre;
        aLeaf.isTemp <- false;
        aLeaf.isIdentifier <- false;
        aLeaf.isRoot <- false;
}}

rule    IdentityAttribute2Identifier {
from    anAttribute : MMDatosTemporal!IdentityAttribute
to      anId:MMGrafo!Identifier(
                name <- anAttribute.name
        )
}

-- Es una regla matched, dada una entidad, la convierte en un nodo raiz del grafo de atributos(Node)
rule  Entity2Vertex {
from  anEntity : MMDatosTemporal!Entity
to    aVertex : MMGrafo!Node ( label <-anEntity.name,isRoot <- anEntity.isRoot,
-- Guardo los identificadores
identifiers <- anEntity.identityAttributes() -> collect( anIdAttr | thisModule.resolveTemp(
anIdAttr,'anId') ),
-- Los atributos descriptivos se convierten en hojas
childrens <- (anEntity.descriptiveAttributes() -> collect(anAttr |
thisModule.resolveTemp(anAttr,'aLeaf')))->
union( anEntity.connections() -> collect(x| thisModule.resolveTemp(x.getEntity())))
}
```

3.3.2 Transformación del Grafo de Atributos al Modelo Multidimensional Temporal

```
module MGrafo2MMultidimensionalTemporal;
create OUT : MMultidimensionalTemporal from IN : MModeloDeGrafo;

helper context MModeloDeGrafo!Node def: verticesNoIdentificadores(): MModeloDeGrafo!Vertex =
           self.childrens -> select (n | not n.isIdentifer);
helper context MMGrafo!Node def: verticesIdentificadores(): MMGrafo!Vertex =
           self.childrens -> select (n | n.isIdentifer);
helper context MMGrafo!Node def: verticesFecha(): MMGrafo!Vertex =
           self.childrens -> select (n | n.label='Fecha');

rule Vertex2Fact {
        from   aRoot : MMGrafo!Node(aRoot.isRoot)
        to      aFact : MMMultidimesionalTemporal!Fact (name<- aRoot.label,
                        dimensions <- Sequence {aRoot.verticesFecha()
                                    -> union(aRoot.verticesIdentificadores())
                                     -> collect(e|thisModule.resolveTemp(e))},
                        attributtes <- Sequence {aRoot.verticesNoIdentificadores()
                                     -> collect(e|thisModule.resolveTemp(e))})}
rule Vertex2Attribute {
        from aLeaf : MMGrafo!Leaf
        to anAtttribute : MMMultidimesionalTemporal!Attributte (name <- aLeaf.label)}

rule Vertex2Dimension {
        from aNode : MMGrafo!Node (not aNode.isRoot)
        to aDimension : MMMultidimesionalTemporal!Dimension ( name<- aNode.label,
            hierarchies <- Sequence{aNode.verticesNoIdentificadores()  ->
            collect(e|thisModule.resolveTemp(e))},attributtes <- Sequence {
           aNode.verticesNoIdentificadores()  -> collect(e|thisModule.resolveTemp(e))})}

rule Vertex2Hierarchy {
        from aNode : MMGrafo!Node
        to aHierarchy : MMMultidimesionalTemporal!Hierarchy (name<- aNode.label)}
```

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 éste 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 [26] 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 [18] 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 y, utilizando sentencias OCL, se establecieron 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 [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 [22] 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 [27] 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 [16] 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 [31] 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 [23] y [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 se transforman 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); 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 diseñar 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 éste al esquema relacional; a continuación, se presentaron los metamodelos del modelo de datos temporal, del grafo de atributos, del modelo multidimensional y del relacional. Finalmente, se presentaron transformaciones formales utilizando ATL. En trabajos futuros se desarrollará un plug-ins en la plataforma Eclipse que permita implementer la

transformación del modelo de datos multidimensional para crear un esquema relacional en diferentes Sistemas Administradores de Base de Datos (SABD).

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

[13] Jouault, F, and Kurtev, I: On the Architectural Alignment of ATL and QVT**.** In: Proceedings of ACM Symposium on Applied Computing (SAC 06), model transformation track, Dijon, Bourgogne, France. 2006

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

[15] 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.

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

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

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

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

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

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

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

[23] 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.

[24] Neil Carlos, Pons Claudia. Aplicando MDA al Diseño de un Datawarehouse Temporal. VII Jornada Iberoamericana de Ingeniería de Software e Ingeniería del Conocimiento. Lima, Perú. 2007.

[25] OCL. Object Constraint Language - version 1.5. 2002.

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

[27] 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.

[28] QVT Partners. Initial Submission for MOF 2.0 Query/Views/Transformations RFP. Version 1.0 (2003.03.03).

[29] The Eclipse Project. http://www.eclipse.org/

[30] UML 2.0 Infrastructure Specification, OMG document ptc/03-09-15, 2003.

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