Design method for a Historical Data Warehouse, explicit valid time in multidimensional models

Método de diseño de un Data Warehouse Histórico, tiempo válido explícito en modelos multidimensionales

Carlos G. Neil[1] Marcelo E. De Vincenzi[1] Claudia F. Pons[1]

Recibido 19 de noviembre de 2012, aceptado 6 de enero de 2014
Received: November 12, 2012 Accepted: January 6, 2014

RESUMEN

Presentamos una nueva estructura de almacenamiento que denominamos DW Histórico y que contiene, explícitamente, el tiempo válido. Nuestra propuesta combina, en un modelo integrado, una Base de Datos Histórica y un DW. El objetivo del modelo propuesto es resolver las limitaciones temporales de las estructuras multidimensionales tradicionales. Aunque el DW Temporal considera, además de la dimensión temporal, otros aspectos vinculados con el tiempo, este modelo solo contempla los cambios que se producen en las dimensiones y jerarquías. Por consiguiente, en virtud de contemplar la necesidad de registrar valores que permitan evaluar tendencias, variaciones, máximos y mínimos, un problema a resolver es cómo registrar, en el diseño de la estructura multidimensional, la variación temporal de los valores de entidades, atributos e interrelaciones, ya que, si bien sabemos que los datos necesarios están almacenados, los mecanismos de búsqueda temporal serían complejos. El diseño del DW Histórico está enmarcado en el enfoque del Desarrollo de Software Dirigido por Modelos, una estrategia que permite, mediante sucesivas transformaciones automáticas, obtener una implementación del modelo multidimensional en un Sistema Administrador de Base de Datos Relacionales (SABDR).

Palabras clave: Método de diseño, desarrollo de software dirigido por modelos, data warehouse, base de datos histórica, tiempo válido.

ABSTRACT

We present a new storage structure that contains the explicit valid time called Historical DW. This proposal combines a Historical Data Base and a DW in one integrated model. The objective of this model is to solve the temporal limitations of the traditional multidimensional structures. Although the Temporal DW considers, besides the temporal dimension, other aspects related to time, this model only takes into account the changes that occur in the DW schema, both in dimensions and in hierarchies. Therefore, considering the need for registering values that allow to evaluate trends, variations, maximum and minimum values, a problem to be solved is how to shape the values of entities, attributes or relationships that may vary in time, in the design of the structure. The fact that the needed data was stored is already known, but the temporal search mechanisms would be complex. The Historical DW design is framed in the MDD approach, a strategy that allows, through successive automatic transformations, to obtain one of the implementations of the model in a Relational Data Base Management System (RDBMS).

Keywords: Design method, model driven software development, data warehouse, historical data base, valid time.

[1] Facultad de Tecnología Informática. Universidad Abierta Interamericana. Buenos Aires, Argentina. E-mail: carlos.neil@uai.edu.ar; medevincenzi@uai.edu.ar; claudia.pons@uai.edu.ar

INTRODUCTION

A Data Warehouse is a "subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions" [10]. A distinctive characteristic of the Data Warehouse is that time is one of the dimensions for the analysis [4, 7], but this refers to the moment when a transaction was made, therefore, it does not specify how or when the values of the entities, attributes and relationships associated with these transactions have varied through time. Although the Temporal Data Warehouse considers, besides the temporal dimension, other aspects related to time [5, 6, 9], this model considers only the changes that occur in the Data Warehouse schema, both in dimensions and in hierarchies. Therefore, a problem to be solved in this type of multidimensional structure, considering the need for registering values that allow to evaluate trends, variations, maximum and minimum values, is how we shape the values of entities, attributes or relationships that may vary in time in the design of the multidimensional structure; since, although the needed data of information was stored, the temporal search mechanisms would be complex [21].

Several methods have been proposed that allow to derive the conceptual multidimensional schema from data sources of the organization and/or of the user's requirements (see [7, 32]); most of them must be done manually [26]. A solution to this problem is proposed by Model Driven Software Development, this approach has become a new paradigm of software development that promises improvements in the software construction based on a model-driven process and supported by powerful tools. This new paradigm aims to improve productivity and software quality generated by reducing the semantic leap between the problem domain and the solution [25].

The proposed design method consists in successive automatic and semiautomatic transformations of models, that begin with an ER model and that finally allow to obtain a temporal multidimensional model expressed as a set of tables in the relational model. The complete proposal includes the automatic creation of a Graphic Query Interface derived from the Historical Data Warehouse that, by means of marks on a graph, automatically creates temporal and decision-making SQL query statements, [23]. The work is complemented with the creation of a prototype based on ECLIPSE technology, which implements the design method of Historical Data Warehouse, the Graphic Query Interface and the automatic execution of SQL statements [35].

To corroborate the proposal empirically, an evaluation of this work was carried out by means of a qualitative research, through a controlled experiment using questionnaires with open-type questions. The research was conducted with students from the Master's Degree in Information Technology from de School of Information Technology of the Universidad Abierta Interamericana. As a first conclusion, from the assessment performance, we consider that the submitted proposal constitutes a valid alternative in the design of multidimensional structures, in the proposal of the design method, and in the storage structure and in the graphical query interface. For more details see [22].

The paper is organized in the following way: firstly, we present the preliminary concepts; secondly, we describe the main characteristics of the Historical Data Warehouse; thirdly, we detail the design method of the Historical Data Warehouse; fourthly, we describe the transformations in informal language; in the fifth place, we present the metamodels used in the transformations; then, we present the automatic transformations; after that, we present the related work; following this, we describe the conclusion and, finally, the future works.

PRELIMINARY CONCEPTS

In this section we will present basic concepts of Data Warehouse, Temporal Data Base and Model Driven Software Development, which allow to support the proposal.

Data Warehouse

The companies use the operational data accumulated over the years and stored in structures *ad hoc* called Data Warehouse to help understand and manage their activities. While an Operational Data Base maintains current data, the Data Warehouse maintains historical data of the company; as a result, the underlying data structures, in order to grow constantly over time require a high storage capacity. Codd [3] introduced the term On Line Analytical Processing (OLAP), in 1993, to characterize the summary requirements,

consolidation, vision and synthesis of data through multiple dimensions. The multidimensional model is the basis of the Data Warehouse. In this model the information is structured in *facts* and dimensions; a *fact* is a topic of interest for the company, it is described by means of *fact* attributes, and these are contained in cells or points in the data cube. A data cube is a multidimensional representation of data that can be seen from different points of view; a data cube is formed by dimensions, which determine the granularity for the representation of *facts* and hierarchies that show how instances of *facts* can be grouped and selected for the decision-making processes [1].

Historical Data Base

In an Operational Data Base, the information becomes effective when it is established and it is considered valid until a new update modifies it; therefore there is no distinction between the time of registration of that information in the Data Base and the period during which the specific values of the facts related to this information are valid in the universe of discourse. Thus, the Data Base represents only the current state and not the history of the reality facts that it was modeling. Furthermore, a Temporal Data Base supports some aspect of time, without taking into account user-defined time [11].

In a Temporal Data Base, an instant is a time point on an underlying time axis; a time interval is the time elapsed betw[een two instants; a timestamp is a temporal mark associated with an object or attribute of the Data Base. The valid time of a fact is the time when the fact is true in the modeled reality. A fact may have any number of instants and time intervals associated, these single instants and intervals are important special cases. Valid times are usually supplied by the user [11]. In practice, the valid time is the most important concept since it models the veracity of the facts recorded in the universe of discourse, which is the main objective of the information systems [20].

A Data Base fact is stored in a Data Base at some point in time, and after it is stored, it is present until logically deleted. The transaction time of a Data Base fact is the time when the fact is present in the Data Base and may be retrieved. Transaction time values cannot be later modified. Also, as it is impossible to change the past, transaction times cannot be changed [11].

The different types of Data Bases are connected, in its definition, to the concepts presented above. A Data Base that models only the valid time is called Historical Data Base, a Data Base that models only the transaction time, is called RollBack Data Base and the one that models both the valid time and transaction time, is called BiTemporal Data Base [11].

Model Driven Architecture

The Model Driven Software Development proposes architecture for the development of computer systems whose goal is to provide a solution for easily adapting the system to changes on business and technology. This approach represents a new paradigm where models of the system, at different levels of abstraction, are used to guide the entire development process. Models are implementation-independent and they are automatically transformed to executable code.

This new paradigm aims to improve productivity and to generate software quality by reducing the semantic leap between the problem domain and the solution [25]. The key underlying idea is that, if working with models, important benefits in interoperability, productivity, portability, maintenance and documentation will be obtained [12]. We can divide, briefly, the Model Driven Software Development process into three phases; in the first one, a Platform Independent Model (PIM) is built, this is a high-level model of the system being developed independently of any technology; then, the previous model is converted into one or more Platform Specific Models (PSM), these models are at a lower level than the PIM and describe the system in accordance with a particular implementation technology; and finally, the last one generates an Implementation Model (IM), this is, a source code from each PSM. The division between PIM and PSM is linked to the concept of platform, which, by not being specifically defined, it does not establish the dividing line between PIM and PSM. Model Driven Software Development also presents a Computation Independent Model (CIM) that describes the system within its environment and shows what is expected from it without showing any details of how it will be built.

The main benefit of the Model Driven Software Development approach is that once every PIM has been developed, we can derive automatically the rest of the models by applying the corresponding transformations in vertical form.

HISTORICAL DATA WAREHOUSE

The Historical Data Warehouse is a new structure of data storage that combines and integrates a Data Warehouse and a Historical Data Base in a single model. This model includes, besides the main analytical *fact*, temporal structures related to the levels of dimensional hierarchies that allow to record data and retrieve information that varies in time.

Conceptual Model

The conceptual model of the Historical Data Warehouse is composed of a *fact* and a set of dimensions; the latter are represented by simple or multiple hierarchical levels (temporal and not temporal) [22].

To express the valid time, we will use the notation [IT, FT) as the representation of the validity interval, where the attribute IT (Initial Time) will be the first instant and the attribute FT (Final Time) the last instant of the described interval. On the other hand, we will consider the interval close/open that will include the instant IT and exclude the instant FT.

The main *fact* may have one or more *fact* attributes, as well as attributes that are not interpreted as *measures* but that will be able to be used to identify a particular instance of the *fact* (degenerated dimensions). Schematically (Figure 1), the model is composed of a *fact* (F) that contains a set of attributes (idi0, i= 1, 2,…, n) that, individually, refers to each of the lowest levels of granularity of the dimensions (ni_0, i= 1, 2,…, m) and, as a whole, this set identifies a particular instance of the *fact*; also the *fact* may contain one or more *measures* (m_i).

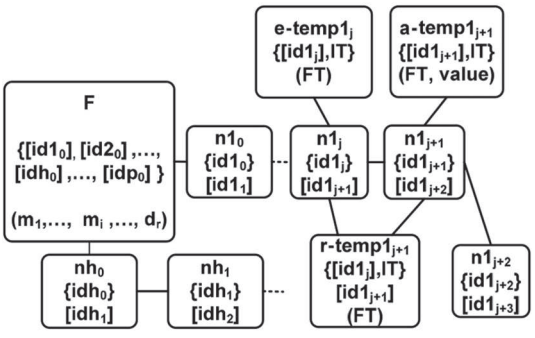

Figure 1. Temporal Multidimensional Model.

Each dimension n_i consists of hierarchical levels (n_{i0}, n_{i1}, $n_{i2,...}$ n_{ij}, i= 1, 2,…, n; j= 1, 2,…, q); all the levels will have an identifier attribute (idn_i) plus an attribute that refers to the greater level of granularity (idn_{i+1}), except for the last level of the hierarchy. Moreover they may have descriptive attributes (non-dimension attributes). The hierarchy levels (e-$temp_i$, a-$temp_i$, r-$temp_i$) represent entities, attributes, and temporal relationships respectively and they constitute non strict hierarchies. The hierarchy level that symbolizes a temporal entity (e-$temp_i$) has a composed attribute that identifies it, formed by the identifier of the level that represents the entity (this identifier will also refer to the entity) plus the attribute called IT ({idn_i, IT}); it also has an attribute denominated FT. The attributes IT (Initial Time) and FT (Final Time) will determine both the initial and final instances of the temporal interval considered respectively. The hierarchy level that represents a temporal attribute (a-$temp_i$) has a compound attribute that identifies it ({idn_i, IT}), consisting of the level identifier that represents the entity (this identifier will also refer to the entity) plus the attribute called IT; the temporal interval also has two attributes denominated FT and *value*. The hierarchy level that represents a temporal relationship (r-$temp_i$) has a compound attribute that identifies it ({idn_i, TI}), formed by the identifier of one of the levels that it links (the identifier will also refer to the level) plus the attribute denominated IT; the temporal relationship also has an attribute that will refer to the other level that links (idn_{i+1}), plus an attribute called FT. In the temporal hierarchies, the attributes IT and FT represent the extreme values of the temporal close-open interval [IT, FT).

Example of a Historical DW

In Figure 5, we showed an example of a Historical Data Warehouse; where both typical queries of a Data Warehouse and queries of a Historical Data Base can be made.

For example, the following are typical queries: At what intervals was a particular client active? On which dates did it change and which was the price of a particular product? Which was the price of a

particular product on a certain date? On what dates and where did a particular client move? Which was the location of a particular client on a certain date?

These are specific queries of a Historical Data Base and could not be performed in a common Data Warehouse. These queries (and the ones of decision-making) can be performed in our model using a Graphical Query interface [23].

Design Method for a Historical DW

The transformation method that begins with an ER model that describes the source data schema of the Operational Data Base, until the obtaining of a Historical Data Warehouse (implemented in an RDBMS) proposes a number of steps, described informally and detailed below. The transformation method will begin with an ER model and, by means of successive transformations, will allow to obtain a set of tables in a Relational Model expressed in SQL sentences.

Below, we will detail how the informal transformation process is; we will show, step by step, each of the transformations in detail. Initially, starting with a Data Model (Figure 2), we will obtain a Temporal Data Model (Figure 3). The Temporal Data Model includes the transformation from the relationship *fact* to an entity *fact*; then, from the previous model we create a Temporal Attribute Graph (Figure 4); afterwards from this one we will obtain a Historical Data Warehouse (Figure 5); from this last one we will model a Relational Model and, finally, we will obtain SQL sentences for its implementation in an RDBMS.

To explain the method, we will use and develop an example that shows the different transformations. Let's present the example: "The Company manufactures and centralizes business operations in Buenos Aires, but its clients are geographically distributed throughout the country. It carries out product sales from customers' orders that are located in different provinces, records transactions performed, the dates and quantities sold".

INFORMAL TRANSFORMATIONS

From Data Model to Temporal Data Model

Below we will detail how to transform the Data Model (Figure 2) to a Temporal Data Model (Figure 3).

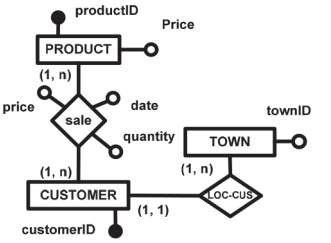

Figure 2. Data Model.

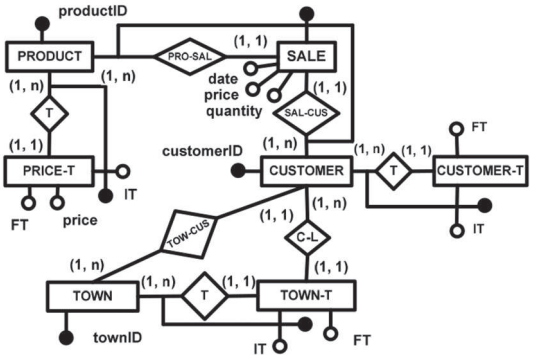

Figure 3. Transformed Data Model.

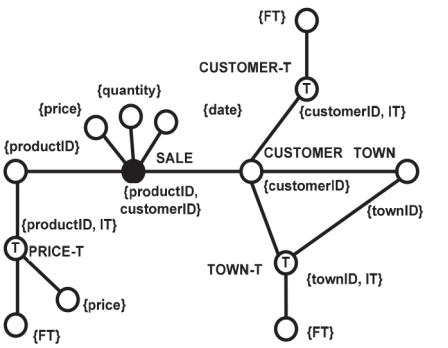

Figure 4. Temporal Attribute Graph.

A temporal entity in the source model will be represented, in the target model, by means of a temporal (weak) entity associated with the (ex) temporal (regular) entity. The new entity will carry the same name as the regular entity and will end in "-T". The temporal entity will contain an attribute called FT and a unique identifier (compound) formed by the identifier of the regular entity plus an attribute called IT; the close-open interval [IT, FT), will represent, now, the lifetime of the temporal entity. The relationship between these two entities will be marked with a "T", the multiplicities of the role of the weak entity will be (1, 1). The multiplicity of the role of the regular entity will be (1, N).

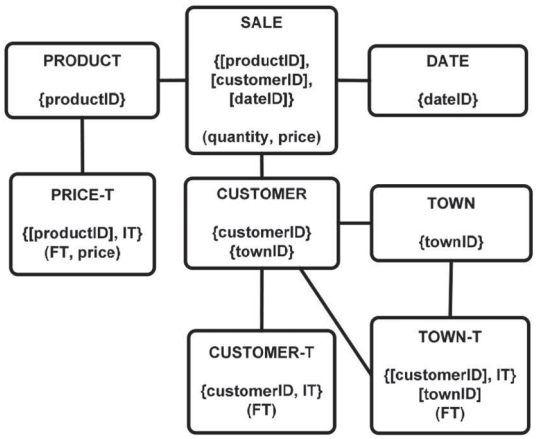

Figure 5. Historical Data Warehouse.

A temporal attribute in the source model will be transformed, in the target model, in a temporal (weak) entity associated with the entity that owns the attribute. The name of the entity will be the same as the name of the temporal attribute and it will end in "-T". The temporal attribute will have as descriptive attributes, the attribute denominated FT and an attribute that will have the same name and domain as the attribute transformed in temporal in the source model. The unique identifier (compound) of the temporal entity will consist in the attribute called IT plus the identifier of the regular entity. The close-open interval [IT, FT), will represent the valid time of the temporal attribute. The multiplicity of the role of the temporal entity will be (1, 1); the multiplicity of the role of the regular entity will be (1, N). The attribute that was transformed to temporal will disappear as such in the regular entity of the new model.

A temporal relationship, in the source model, will transform into a temporal entity (weak) linked to one of the entities that form the relationship in the target model. The name of the new entity will be established as a combination of the name of the elected regular entity and will end in "-T", the multiplicities of the role of the related entities will be (1, N), the multiplicities of the role of the temporal entity will be (1, 1). The new temporal entity will have as descriptive attribute, the attribute denominated FT plus the attributes that the relationship will have in the ER source model. The unique identifier (compound) of the temporal entity will consist in the attribute called IT plus the identifier attribute of the entity that gave the name (it will represent a weak entity of that one). The relationship that became temporal will remain as non-temporal to allow, in the subsequent transformation processes, to be source of a possible level of grouping in the dimension. All the entities and non-temporal relationships, in the source model, will remain without modifications in the target model. All the non-temporal attributes (both entities and relationships), in the source model, will stay in the target model, unlike the attributes transformed in temporal attributes.

The *fact* (if represented by a relationship) in the source model, will become an entity related to the two entities that are part of the relationship in the target model. The name of the entity will be the same as the one of the relationship, the multiplicity of the role of the linked entities will be (1, N), the multiplicities of the role of the transformed entity will be (1, 1). The *fact* will have the attributes of the relationship (if any) as descriptive attributes and the union of the identifier attributes related to the relationship as a unique identifier.

From Temporal Data Model to Temporal Attribute Graph

As an intermediate step in the design of the Historical Data Warehouse, a modified Temporal Attribute Graph [7] will be built, from the Temporal Data Model (Figure 3), to be used, later, as a source model for the construction of the Historical Data Warehouse.

For the construction of the Temporal Attribute Graph (Figure 4), the main *fact,* in the Temporal Data Model, will be identified, first, as a concept of primary interest for the decision-making process in the Historical Data Warehouse. The *fact* will correspond to events that occur dynamically in the reality and will be represented, in the adapted model, by an entity. The main *fact* will be composed of vertexes, each one will represent an attribute, that may be simple or compound and that will belong to the source model.

We will detail the characteristics of the graph: the *root* of the graph will correspond to the identifier of the *fact* in the model. For each non-temporal vertex *v*, its corresponding associated attributes will determine functionally to all attributes that apply to the descendants of *v*. Each vertex *v* labeled with a "T" will correspond to a temporal vertex, this in turn, can be "terminal" or "non-terminal", and

the first one will be derived from an attribute or a temporal entity in the model whereas the second one will be derived, from a temporal relationship. The vertexes "Terminal" do not have descendant vertexes whose attributes are identifiers. The vertexes "non-Terminal" has descendant vertexes whose attributes are identifiers.

Construction of a Temporal Attribute Graph

Given an identifier (E) that indicates a set of attributes that identify the entity E in the source model, the Temporal Attribute Graph will be able to be constructed automatically by applying the following recursive function [22].

```
Function translate (E: Entity): Vertex
{
      v = newVertex(E);
// newVertex(E) creates a new vertex, containing
// the name and the identifier of the object E
      for each attribute a ∈ E | a ∉ identifier(E) do
      addChild (v, newVertex(a));
//adding a child "a" to the vertex v
      for each entity G connected to E
      by relationship R | mult-max(E,R)=1 or
      (R is temporal and E is not temporal) do
//are considered temporal entities
//and infinite loops are avoid
      {for each attribute b ∈ R do addChild (v,
      translate(G)); }
      return(v)
}
```

From Temporal Attribute Graph to Multidimensional Model

The transformation process of the Temporal Attribute Graph to the Temporal Multidimensional Model (Figure 1), implies the choice of which vertexes of the graph will be considered *measures* in the *fact* and which dimensions or levels of hierarchies (temporal or not). All of them depend on the designer's decisions. Unlike the choice of the *fact*, that will directly derive from the *root* of the graph. The identifier attributes of the nodes in the graph will be preserved in the Temporal Multidimensional Model for the subsequent transformation to the Relational Model. The concepts of *root*, *leaf*, *child* and *parent* were taken from [19].

The general approach that we will use in the transformation is as follows: The *fact* of the graph in the Temporal Attribute Graph will correspond to the *fact* in the Temporal Multidimensional Model; its name will be the same as the one of the *root*. The identifier attributes in the *root* node will be identifier attributes in the *fact*.

All the vertexes linked to the *root* in the Temporal Attribute Graph, which are neither identifiers nor attributes that denote temporal aspects (the latter will be transformed in leaves of the temporal hierarchy level) will correspond to *measures*. A *fact* may lack *measures*.

The *leaf* levels in the hierarchy should be chosen in the Temporal Attribute Graph between vertexes *children* of the *root*. Their choice will be crucial for the design of the Historical Data Warehouse since they will determine the lowest level of granularity of the instances of the *fact*. In the Temporal Attribute Graph, the vertexes linked to the *fact*, that will not be identifiers, will be the *leaves* in the Temporal Multidimensional Model; the vertexes associated with them, which will not be identifiers, will be non dimension attributes, the name of the vertex will be the same as the one of the *leaf*. The multiplicity of the role *fact* will be (1, N), the multiplicity of the role *leaf* will be (1, 1). The *leaf* levels in the hierarchy are linked to the main *fact*; therefore, each one of the attribute identifiers of the *fact* will be a reference to each one of the *leaf* hierarchy levels, so as to establish a "many-to-1" between the *fact* and *leaf* levels.

The *fact* should have an associated vertex that denotes temporal aspects. It will become *leaf* temporal of the time dimension in the Temporal Multidimensional Model, if this is not the case, because there is not an attribute of these characteristics in the Temporal ER model, the *leaf* temporal will be directly forced in the Temporal Multidimensional Model and the temporal dimension identifier will be added to the *fact* identifier. The multiplicity of the role *fact* will be (1, N), the multiplicity of the temporal role *leaf* will be (1, 1).

The vertexes linked to the adjacent nodes to the *root*, in the Temporal Attribute Graph, which are identifiers, will be transformed in *parent* levels of the hierarchy into the Temporal Multidimensional Model. The name of the *parent* level will be the same as the one of the vertex and its identifier will

be the same as the identifier of the node, all the vertexes linked to that level, which are identifiers, will also be linked to *parent* levels. The vertexes that are not identifiers will be level attributes. All the *child* levels in the hierarchy will have an attribute that will make reference at the *parent* level. In all the levels of the hierarchy, the multiplicity of role of the *parent* will be (1, 1), the multiplicity of *child* role will be (1, N).

The temporal vertexes (marked with "T"), in the Temporal Attribute Graph, linked to the vertexes transformed into levels of the hierarchy in the Temporal Multidimensional Model will be transformed in temporal levels, the level name will be the same as the one of the vertex and the identifier will be the same as the identifier of the node. If the vertex was "Terminal", all the linked vertexes, will be *leaves* and, therefore, will not be identifiers, they will be attributes of the temporal hierarchy in the Temporal Multidimensional Model. Besides, the attribute that is not part of the temporal interval, will make reference to the linked hierarchy level. If the vertex was "non-Terminal", it would be linked to the two non-temporal hierarchies described in the graph; therefore, the temporal level will have references to these two levels. All the vertexes linked to the temporal nodes that are not identifiers, will be temporal vertex attributes.

From Temporal Multidimensional Model to Relational Model

The next step in the transformation is the creation of a Relational Model. Therefore, from the Temporal Multidimensional Model we will obtain a set of relational data structures by applying the following transformation rules. The transformation process will be detailed below.

The representation of the *fact* in the Temporal Multidimensional Model, will be transformed into a table in the Relational Model; the *measures* will be the columns of the table; the primary key will be composed of a set of the identifier attributes of the *leaf* levels; the attributes forming the primary key will also be foreign keys that will refer to each one of the resulting tables from the transformation of *leaf* levels associated to the *fact*.

The *leaf* levels will become tables in the Relational Model; the attributes will be columns of the table, the primary key will be composed of a group of the identifier attributes of each level; in addition, each table *leaf* will have a foreign key that will refer to each one of the tables that form the hierarchy levels.

The levels in the hierarchies in the Temporal Multidimensional Model will become tables in the Relational Model; the level attributes will be columns of the table; the primary key will be composed by the group of the identifier attributes of the level; also, each table *child* will have a foreign key that will refer to each one of the linked tables *parent*.

The temporal hierarchy levels in the Temporal Multidimensional Model will become tables in the Relational Model. If this becomes from a temporal entity, it will have the FT as an attribute; the primary key will be composed of the union of the primary key of the related hierarchy table (in addition, it will be the foreign key that will refer to this hierarchy table) plus IT attribute. If the temporal hierarchy is derived from a temporal attribute, it will have the attribute FT plus the attribute that is necessary to conserve temporarily as attributes; the primary key will be composed of the union of the primary key of the related hierarchy table (in addition, it will be the foreign key that will refer to this hierarchy table) plus the attribute IT. If the temporal hierarchy level comes from a temporal relationship, it will have as an attribute the FT and the attribute that represents the primary key of one of the related hierarchy tables (also, it will be the foreign key that will refer to the hierarchy table); the primary key will be composed of the union of the primary key of the other related hierarchy table (in addition, it will be the foreign key that will refer to this hierarchy table) plus the attribute IT.

From Relational Model to Tables

The transformation of the Relational Model to the Data Model expressed in SQL sentences is immediate. Each attribute has a data type, which will have to be transformed (using a defined conversion table) to existing SQL data types. Below, the *Pseudotables* from the Relational Model transformed in the previous section are shown:

SALE(*productID(PRODUCT),*
customerID(CUSTOMER), date, quantity, price)
PRODUCT(*productID, …)*
CUSTOMER(*customerID,…)*

CUSTOMER-T(*{customerID (CUSTOMER), IT}, FT)*
TOWN(*townID,…)*
PRICE-T(*productID(PRODUCT), IT, FT, price)*
TOWN-T(*{CUSTOMER(CUSTOMER-T), IT}, FT, townID(TOWN))*

METAMODELS

A metamodel is a mechanism that allows to define modeling languages formally. Therefore, a metamodel of a language (graphical or textual) is a precise definition of its elements by using concepts and rules of a certain meta-language needed to describe models in that language.

The Meta Object Facility (MOF) [13] defines a common and abstract language to define modeling languages and the form to access and exchange models expressed in these languages. There are two fundamental reasons for using metamodels in the Model Driven Software Development framework: firstly, the need of a mechanism for defining modeling languages that are not ambiguous, so that a transformation tool can read, write, and understand the models. Therefore, in the Model Driven Software Development, models are defined by metamodels. Secondly, the need of transformation rules that describe the definition of this transformation and details of how a model, in a source language, could be transformed into a model in a target language, using the source and target metamodels to define the transformation, so that they are defined in general and not for a particular application.

In our proposal, the approach used is the Model Driven Software Development, in particular, we create Domain-Specific Models (DSM) using a focused and specific language for each one of them (DSL). In this case, we do not use CWM but simpler metamodels, instances of MOF, for Data Models, the Multidimensional Model and the Relational Model. In addition, we designed specific metamodels for the models used in the process: for the construction of the Temporal Attribute Graph, we use the metamodel Temporal Attribute Graph. We do not use UML or profiles for the design of PIM, as we believe that the ER model is more expressive for data modeling.

Next, we will present the metamodels used for the transformations. All the classes, except those of the Temporal Attribute Graph, inherit the name attribute of a superclass *Named*, not shown in the graphs.

The Data Metamodel

The Data Metamodel (Figure 6) will be used to make the horizontal transformation (PIM to PIM), of the basic Data Model to the Temporal Data Model (Figure 7). This process will allow to transform the entities, the attributes and the temporal relationships in the source model, in temporal entities in the target model and, also, the relationships *fact*, in entities.

Temporal Data Metamodel

The Temporal Data metamodel (Figure 7) will be used together with the Temporal Attribute Graph metamodel (Figure 8) to make the horizontal

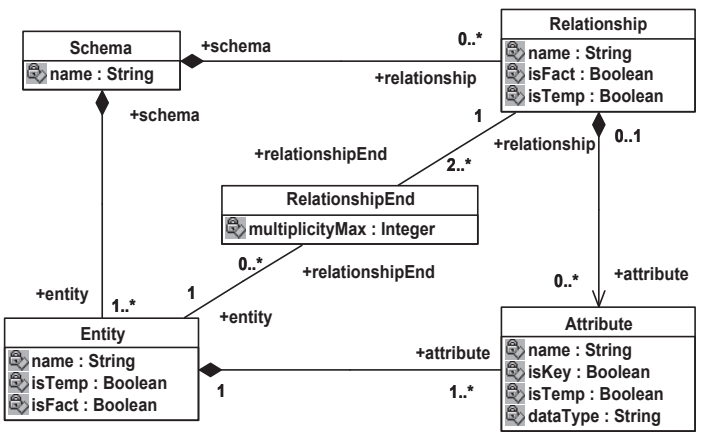

Figure 6. Data Metamodel.

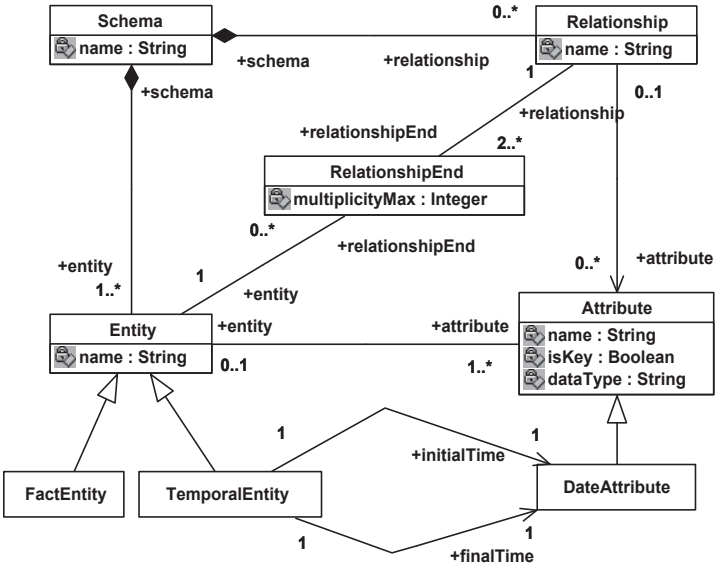

Figure 7. Temporal Data Metamodel.

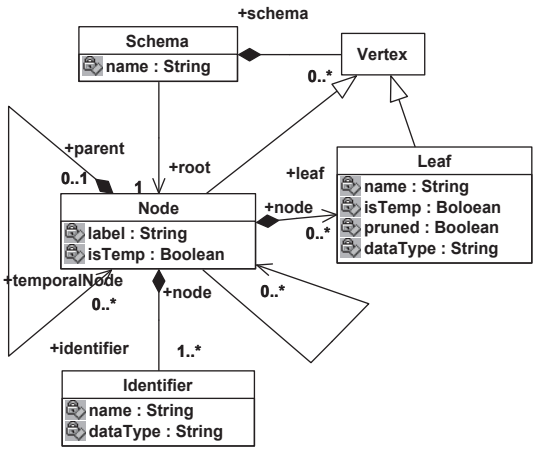

Figure 8. Temporal Attribute Graph Metamodel.

transformation (PIM to PIM), from the Temporal Data Model to the Temporal Attribute Graph.

Attribute Graph Metamodel

The Temporal Attribute Graph Metamodel (Figure 8) will be used, together with the Temporal Data Metamodel (Figure 7), for horizontal transformation (PIM to PIM) of the Temporal Data Model adapted to the Temporal Attribute Graph.

Relational Metamodel

The Relational Metamodel (Figure 10) will be used, together with the Temporal Multidimensional Metamodel (Figure 9), in the PIM to PSM transformation of the Temporal Multidimensional Model to the Relational Model.

AUTOMATIC TRANSFORMATIONS

The model transformations are the main component in the Model Driven Software Development. For the transformations from model to model we use ATL [2] that is a hybrid
programming language (imperative and declarative).

The declarative style is recommended, since it allows the mappings between the source and target model in a simpler way. ATL imperative constructions are used to describe mappings that are difficult to specify in a declarative style.

An ATL transformation is composed of rules that define how elements in the source model are used to create and initialize the elements of the target model. ATL is supported by a tool built on the Eclipse platform that facilitates the development of ATL transformations.

For implementing the transformations from model to text we use MOFScript. This tool assists in the Model Driven Software Development process by supporting the source code generation and other kinds of text generation from the models, such as documentation generation.

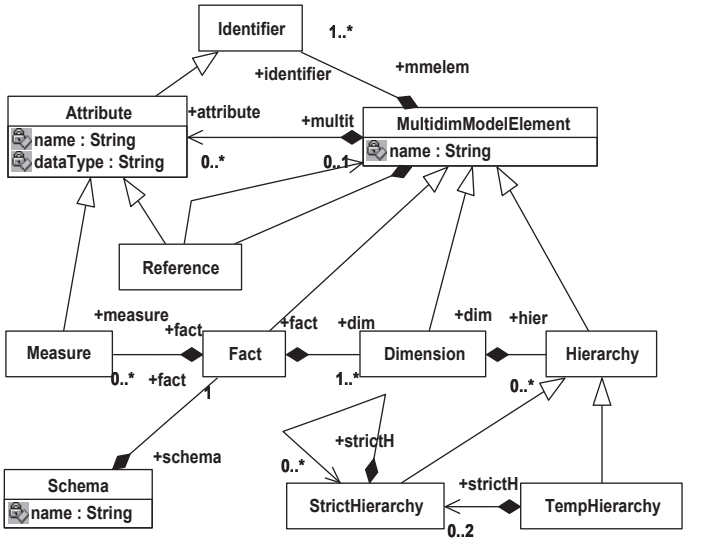

Figure 9. Temporal Multidimensional Metamodel

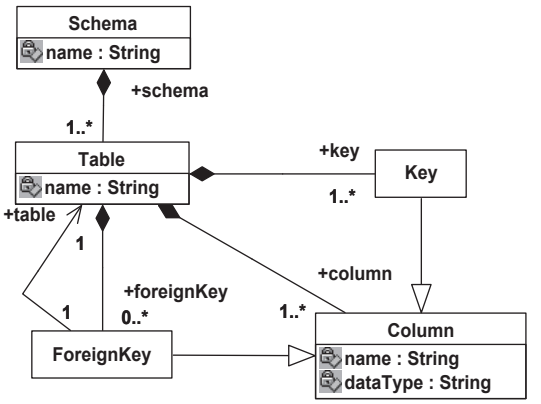

Figure 10. Relational Metamodel.

MOFScript has few constructions. It is easy to use and understand, and has a similar style to the scripting languages. It is a language that is influenced by MOF QVT, in particular, MOFScript specializes QVT [14]. For space limitations, the ATL and MOFScript code of the transformations is not included in this paper, but can be downloaded from [35].

RELATED WORK

In the last years, there have been several studies related to the use of the Model Driven Software Development approach for the design of computer systems; in particular, proposals were presented using the Model Driven Architecture (MDA) for the design of different types of storage structures, such as Temporal Data Base, Data Warehouse and Spatial Data Warehouse. In [16] a framework for the development of a hybrid multidimensional model by using a conceptual representation that can automatically derive into logic model was presented. In [8] an MDA approach for the design of a Spatial Data Warehouse was presented. In [24] an MDA approach for the transformation of a temporal data model to a relational schema was proposed. In [15] an approach based on model driven reverse engineering for the development of Data Warehouse was presented. In [33] the development of an ORDB in the framework of MDA was presented.

Transformations are defined to generate the schema of the Data Base (PSM), starting from the conceptual data schema (PIM) represented by UML class diagrams. In [29] an extension of the relational package CWM to represent, at a logical level, all security and audit requirements captured during the conceptual modeling phase of the Data Warehouse was proposed. In [27-28, 30-31], with similar approaches, a set of MDA transformations through the QVT standard for transforming a multidimensional secure conceptual model into logical secure relational schema was presented. In [22], in the framework MDA, a set of transformations to derive into Temporal Multidimensional Model from a data model with temporal marks was proposed. They presented a semi-automatic methodology for generating a relational schema of a Temporal

Data Warehouse from a temporal data model. In [17] an approach to ensure the correctness of a conceptual Data Warehouse from the data sources that fill it was presented. They obtained a conceptual Multidimensional schema of the Data Warehouse from the user's requirements. In [34] a semi-automatic methodology for the conceptual design of a Data Warehouse was presented. In [18] a framework, oriented to MDA, for the development of a Data Warehouse was presented. The proposed framework has the complete design of a Data Warehouse as objective, where they align each one of the development stages with different points of view of MDA. In this paper they presented the MD2A, an approach that applies MDA for the development of a Data Warehouse; they defined an MD PIM, an MD PSM and the corresponding transformations using QVT language. The PIM is modeled using UML profiles and, the PSM, using the CWM relational package.

On the other hand, the works related to the design of data structures using MDA approach presented by other authors consider the following objectives: to improve the productivity in the development of a Data Warehouse, in the MDA framework ([15, 16, 17, 18, 34]); to use the MDA approach in the design of a Spatial Data Warehouse [8]; to consider safety issues in the Data Warehouse ([27, 28, 30, 31]) and to implement them in a specific OLAP tool or, finally, to use the MDA approach for the development of an ORDB [33]. The approach used in all cases by the presented works is in the MDA framework; this involves the use of associated standards proposed by the OMG (UML and profiles, OCL, XMI, CWM, and QVT).

Our proposal differs from the referenced works, mainly in the Multidimensional Model proposed: the Historical Data Warehouse represents a new data structure that combines and integrates, in a single model, a Data Warehouse and a Historical Data Base.

CONCLUSIONS

The main proposal of the work is the creation of a model and a method for the automatic design of a Historical Data Warehouse, that is, a new data storage structure that combines and integrates a Data Warehouse and a Historical Data Base, in a single model. This Temporal Multidimensional Model includes, besides the main analytical *fact*, temporal structures related to the levels of the dimensional hierarchies that make possible to record the data and to retrieve information that varies in time. Using the Model Driven Software Development paradigm, the Historical Data Warehouse is generated from a design method that, using a conceptual data model as source expressed in an ER model and by successive transformations, allows to obtain a logical implementation in an RDBMS.

Our proposal differs from other related works in several aspects. First, the proposed storage structure (Historical Data Warehouse) is original in relation to other works, where the main focus is centered on Data Warehouse, Spatial Data Warehouse and ORDBs; second, the approach used in this work is the Model Driven Software Development, in particular DSM, and our proposal uses DSL. Unlike the MDA approach, where they promote the use of OMG standards, we do not use UML or profiles for the design of PIM. This simplifies the designer's work, since he should only learn a simple notation, with limited elements and focused on their expertise domain.

In this regard, the applicability of the Model Driven Software Development paradigm is important to emphasize that most of the presented papers are connected with storage structures. Undoubtedly, the main cause is the relative simplicity for representing the transformation of static structures, unlike the dynamic part (the functional aspects) of the computer systems whose essence is more complex to capture. Regarding this last point, our proposal considers aspects related to the behavior, in some way, by permitting to derive queries on the Data Base.

FUTURE WORK

From the work presented, a range of possible lines of research is opened that were not considered in the development of the work but which deserve to be taken into account in future works. Below, we will detail, the issues that we have not considered and whose solution involves a line of research to develop:

The creation of a DLS for Data Warehouse transformations. In our work we used the ATL

language to define transformations between the models proposed. ATL is a hybrid model transformation language that allows both declarative and imperative constructs to be used in transformation definitions, that is, it provides syntactic constructions focused on the definition of transformations among models. Additionally, these transformation languages (LT) can accept a higher degree of specialization, that is, we could define a domain-specific language for the transformation. In our case, we could define specific LT Data Warehouse transformations. To have a specific language versus a more general language, such as ATL, would remarkably facilitate the definition and reuse of transformations.

The consideration of the user's requirements in the design of the Historical Data Warehouse. Our work uses a conceptual data model expressed in an ER model, as the source model for the transformation process, which we believe represents the information requirements of the users, at least in regard to the transactional application. We have not considered how to evaluate and capture the information requirements of the users in multidimensional and temporal aspects in the Historical Data Warehouse design. Partly, it is that way because the transformations begin with the PIM, regardless of the CIM, explicitly. The transformation from CIM to PIM is not a very developed area, opening a line of research to be considered in future works.

The extension of the data model. The temporal data model used is the ER standard model, which only includes the basic constructions and, through them, the capture of the temporal aspects implicitly. We have not initially considered grade relationships in our data model > 2 and neither did we extend its semantic concepts such as generalization, aggregation and temporal constructions. These aspects deserve to be evaluated and considered in future extensions of the data model used.

The temporal integrity restrictions. The model, as it is planned, does not consider restrictions in regard to the updates, so the case of temporal overlapping may exist. The establishment of restrictions regarding the insertion, deletion and modification of temporal values should prevent possible inconsistencies in the Data Base.

The automatic derivation of the ETL process. Another aspect that we have not considered in our Historical Data Warehouse design is how to perform the ETL process. This process is important because it is responsible for integrating data from different heterogeneous sources. For the construction of the conceptual model we start from an ER model that represents a non-historical Data Base. The extension to a temporal model does not conceptually imply greater inconvenience. The load of historical data does require considering strategies for the fill up of the Historical Data Warehouse from data coming from backups stored in different supports and formats. One line of research to consider is, in the context of the Model Driven Software Development, the automatic code generation of ETL processes.

The use of an Object Relational PSM. We have used the relational model, in particular the standard SQL92 for the development of PSM. The ORDB Data Bases (standard SQL2003) presents such constructions as the abstract data types defined by the user, which would permit a simpler representation of the proposed temporal model, allowing to develop an associated line of research.

REFERENCES

[1] R. Agrawal, A. Gupta and S. Sarawagi. "Modeling multidimensional databases". Research Report. IBM Almaden Research Center. San Jose, California, EE.UU. 1995.

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

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

[4] S. Chaudhuri and U. Dayal. "An overview of data warehousing and OLAP technology". ACM SIGMOD Record. Vol. 26, Issue 1, pp. 65-74. March, 1997.

[5] J. Eder and C. Concilia. "Evolution of dimension data in temporal data warehouses". Technical Report. 2000.

[6] J. Eder, C. Concilia and T. Morzy. "A model for a temporal data warehouse". Proc. of the

Int. OESSEO 2001 Conference. Rome, Italy. 2001.

[7] M. Golfarelli, D. Maio and S. Rizzi. "Conceptual design of data warehouses from E/R schemes". Proceedings 31st Hawaii International Conference on System Sciences. Hawaii, USA. 1998.

[8] O. Glorio and J. Trujillo. "An MDA approach for the development of spatial data warehouses". DaWaK. Turin, Italy. 2008.

[9] C. Hurtado, A. Mendelzon and A. Vaisman. "Maintaining data cubes under dimension updates". Proc. of the 15th Int. Conf. on Data Engineering. Sydney, Australia. 1999.

[10] W.H. Inmon. "Building the Data Warehouse". John Willey, pp. 576. 2005.

[11] C.S. Jensen. "A consensus glossary of temporal database concepts". ACM SIGMOD Record. Vol. 23, Issue 1, pp. 52-65. March, 1994.

[12] A.G. Kleppe, J. Warmer and W. Bast. "MDA Explained: The model driven architecture: practice and promise". Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 2003.

[13] OMG. "MOF Meta Object Facility Specification". OMG Document formal. URL: http://www.omg.org

[14] OMG. "MOF 2.0 Query/View/Transformations OMG Adopted Specification". March, 2005. URL: http://www.omg.org

[15] J.N. Mazón, E. Ortega y J. Trujillo. "Ingeniería inversa dirigida por modelos para el diseño de almacenes de datos". En: XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD). España. 2007.

[16] J.N. Mazon and J. Trujillo. "A hybrid model driven development framework for the multidimensional modeling of data warehouses". SIGMOD Record. Vol. 38, Issue 2. June, 2009.

[17] J.N. Mazon, J. Trujillo and J. Lechtenbörger. "A set of QVT relations to assure the correctness of data warehouses by using multidimensional normal forms". ER 2006. Tucson, USA. 2006.

[18] J.N. Mazon, J. Trujillo, M. Serrano and M. Piattini. "Applying MDA to the development of data warehouses". DOLAP 2005. Bremen, Germany. 2005.

[19] E. Malinowski and E. Zimányi. "Hierarchies in a multidimensional model: from conceptual modeling to logical representation". Data & Knowledge Engineering. Vol. 59, Issue 2, pp. 348-377. November, 2005.

[20] P. Mc Brien, A. Seltveit and B. Wangler. "An Entity-Relationship model extended to describe historical information". CISMOD, pp. 244-260. Bangalore, India. July, 1992.

[21] C. Neil and J. Ale. "A conceptual design for temporal Data warehouse". 31º JAIIO. Simposio Argentino de Ingeniería de Software. Santa Fe, Argentina. 2002.

[22] C. Neil. "Diseño de un almacén de datos histórico en el marco del desarrollo de software dirigido por modelos". Tesis para optar al grado de Doctor en Ciencias Informáticas. Universidad Nacional de La Plata. La Plata, Argentina. 2010.

[23] C. Neil, J. Irazábal, M. De Vincenzi and C. Pons. "Graphical query mechanism for historical DW within MDD". XXIX Conferencia Internacional de la Sociedad Chilena de Ciencia de la Computación. IEEE Press. Curicó, Chile. 2010

[24] C. Neil y C. Pons. "Aplicando QVT en la transformación de un modelo de datos temporal". Jornadas Chilenas de Computación. Punta Arenas, Chile. 2008.

[25] C. Pons, R. Giandini y G. Pérez. "Desarrollo de software dirigido por modelos. conceptos teóricos y su aplicación práctica". McGraw-Hill Education, p. 300. Buenos Aires, Argentina. 2009.

[26] O. Romero and A. Abelló. "MDBE: automatic multidimensional modeling". ER 2008. California, USA. 2008.

[27] E. Soler, J. Trujillo, E. Fernández-Medina and M. Piattini. "A set of QVT relations to transform PIM to PSM in the design of secure data warehouses". ARES 2007. Viena, Austria. 2007.

[28] E. Soler, J. Trujillo, E. Fernández-Medina and M. Piattini. "Designing secure data warehouses by using MDA and QVT". Journal of Universal Computer Science. Vol. 15, Issue 8, pp. 1607-164. 2009.

[29] E. Soler, J. Trujillo, E. Fernández-Medina and M. Piattini. "Una extensión del metamodelo relacional de CWM para representar almacenes de datos seguros a nivel lógico". JISBD 2007. Zaragoza, España. 2007.

[30] E. Soler, J. Trujillo, E. Fernández-Medina y M. Piattini. "Aplicación de QVT al desarrollo

de almacenes de datos seguros: un caso de estudio". IDEAS 2007. Isla Margarita, Venezuela. 2007.

[31] E. Soler, J. Trujillo, E. Fernández-Medina y M. Piattini. "Un conjunto de transformaciones QVT para el modelado de almacenes de datos seguros". JISBD 2007. Zaragoza, España. 2007.

[32] N. Tryfona, F. Busborg and J.G.B. Christiansen. "starER: a conceptual model for data warehouse design". In: ACM Second International Workshop on Data Warehousing and OLAP (DOLAP'99). Missouri, USA. November, 1999.

[33] J.M. Vara, B. Vela, J.M Cavero y E. Marcos. "Transformación de modelos para el desarrollo de base de datos objeto-relacionales". IEEE Latin American Transactions. Vol. 5 N° 4. Julio 2007.

[34] L. Zepeda y M. Celma. "Aplicando MDA al diseño conceptual de almacenes de datos". 9º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS'06). Mar del Plata, Argentina. 2006.

[35] C. Neil. "Diseño de un Almacén de Datos Histórico en el Marco del Desarrollo de Software Dirigido por Modelos". 19 de diciembre de 2013. URL: http://www.lifia.info.unlp.edu.ar/eclipse/pages/tesis_neil.htm