Graphical Query Mechanism for Historical Data Warehouse within MDD

Carlos Neil, Jerónimo Irazábal, Marcelo De Vincenzi, Claudia Pons

Universidad Abierta Interamericana (UAI)
Ciudad Autónoma de Buenos Aires. Argentina
[carlos.neil, medevincenzi, claudia.pons]@uai.edu.ar
jirazabal@lifia.info.unlp.edu.ar

Abstract

Query tools that depend on the ability of programmers, impose a cognitive load that could reduce the user's productivity. Within MDD, the proposal of our work is the creation of a visual query mechanism derived from a temporal multidimensional data structure. This mechanism facilitates (and partially automates) the formulation of temporal and decision making queries.

1. Introduction

A distinguishing characteristic of the OLAP tools is its ability for grouping the measures by means of the election of one or more dimensions; the multidimensional visualization makes it possible to see the different levels of the hierarchies in each dimension in a logical way, allowing the user to get a more intuitive understanding of the data. The existing commercial software provides different views of the multidimensional elements. In order to facilitate the queries, a graphical language would have to operate on an explicit graphical view of the conceptual multidimensional scheme; in the same way, the OLAP queries would have to be expressed using a graphical representation in an incremental form [21].

A temporal Data Base (DB) contains present, historical and future data. There exist several application domains where it is necessary to access not only to the most recent state of the data base, but also to last and even to future states. Most of the temporal DBs lack interfaces centered in the user.

The specification of queries on a database structure represents a big challenge for non-technical users. The traditional form to access a data base has been by means of SQL, a textual language designed specifically to create, organize and consult data bases. Due to the complexity of the formulation of non-trivial searches, a number of approaches have been proposed with the goal of making them more accessible to a wider range of users [8]; the use of graphical languages, compared with the algebraic expressions, facilitates, to the end user, the specification of queries [21].

A user-centered visual environment should include tools for modeling the temporal data structure as well as for the formulation of queries. Such tools should also support different visualization metaphors for the addition and updating of temporal data [12].

Queries become even more complex when the data is stored into intricate structures such as a historical Data Warehouse (DW) [4]. One way to address this complexity is through the use of the Model Driven Software Development (MDD) approach [18]. The MDD proposes 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.

The aim of our work is to provide a simplification in the information searching mechanism on Historical DW, where the users can carry out the queries by making use of a graphical environment, without considering any implementation detail. In this article we present: 1) the development of a visual environment for making queries on temporal multidimensional structures that is automatically derived from the historical DW by applying MDD principles 2) the automatic generation of SQL statements, using the MDD approach, that allows you to perform complex queries on a historical DW as well as typical queries on a historical DB 3) the creation of a prototype, based on Eclipse, that implements the graphical interface for query and the evaluation of SQL statements.

The rest of the work is structured as follows: section 2 presents the basic historical database concepts; section 3 explains the most important features of historical DW; section 4 describes the main principles of MDD; section 5 describes the graphical queries and how they are resolved; section 6 presents the informal transformations that allow the creation of the query
2. Historical DB

Temporal information can have different semantics depending on the considered temporal dimension. A temporal DB usually defines one or two of the following temporal dimensions: valid time and transaction time. The valid time of a fact is that in which the fact is, was or will be true. The transaction time of a fact is one in which that fact is up to date in the data base and can be recovered. An entity lifespan captures the time of existence of that entity. Lifespan of an entity “E” can be modeled as the valid time of the fact “E’ exists” [7]. Different types of databases are related, in its definition, to the concepts previously presented. A database that only models the valid time of entities is called Historical DB [9].

3. Historical Data Warehouse

A main characteristic of the DW is that time is one of the dimensions for analysis [4] [7]; time refers to the instant in that a transaction was performed. As a consequence, a problem to be solved in this type of structure - in view of the need to register values to assess trends, variations, upper and lower bounds - is how the expected variation through the time of the values of the entities, attributes or interrelations impacts on the design of the data structure [15]. The proposed solution is the design of a complex structure named Historical DW that combines, in an integrated model, a Historical Data Base and a Data Warehouse [15]. The Historical DW is composed of a fact and a set of dimensions; the later are represented by levels of hierarchies (temporal and non-temporal) that can be either single or multiple. The main fact can have one or more fact attributes (or measures), apart from other attributes that are not interpreted as such but can be used to identify a particular instance of the fact (degenerate dimensions).

Schematically (Figure 1), the model is composed of a fact (H) that contains a set of attributes (id) which, individually, make reference to each leaf level in the dimensions (ni), and together, identify a particular instance of the fact; Furthermore, the fact can contain one or more measures (medi).

Each dimension ni (i, i = 1, 2, ..., n) consists of hierarchies levels ni (i, i = 1, 2, ..., n; j, j = 1, 2, ..., m); all levels have an ID attribute (idn) and an attribute that refers to the level father (idn+1), with the exception of the root level, being the last level of the hierarchy; they may also have descriptive attributes (no-dimension attributes). The levels of hierarchy (e-temp, a-temp, r-temp), representing entities, attributes, and temporal interrelations respectively, are not strict hierarchies. The hierarchy level that symbolizes a temporal entity (e-temp) has a composite attribute that identifies it, formed by the identifier of the level that represents the entity (it refers to it) and the attribute named TI ((idn, TI)); in addition, it has an attribute called TF. The level of hierarchy that represents a temporal attribute (a-temp) has a composite attribute that identifies itself ((idn, TI)), formed by the identifier of the level that represents the entity (it refers to it) plus an attribute called TI; in addition it has two attributes called TF and value. The hierarchy level that represents a temporal relationship (r-temp) is identified by a composite attribute ((idn, TI)); this attribute is formed by the ID of one of the linked levels plus the attribute named TI; it also has an attribute that references to the other connected level (idn+1), plus an attribute called TF. In the temporal hierarchies, the attributes TF and TI represent the upper and lower bounds of of the time interval [TI, TF].

Figure 1. Temporal Multidimensional Model
4. Model Driven Software Development

The MDD proposes an 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. In this way, significant benefits in productivity, portability, interoperability, maintenance, and documentation are obtained.

The MDD process can be divided into three phases; the first constructs a platform independent model (PIM), this is a high-level technology-independent model; then, the previous model is transformed into one or more platform specific models (PSM); these models are lower level and describes the system in accordance with a given deployment technology; finally, the source code is generated from each PSM. Additionally, MDD describes a model that is independent of computational aspects (CIM), which describes the system within its environment and shows what behavior is expected without exposing details of how it will be built. The main benefit of the MDD approach is that once each PIM has been developed we automatically derive the rest of the models by applying the corresponding transformations in vertical form.

5. Queries on a historical DW

It is a challenge for non-technical users to specify queries on a complex database structure of [12]. Due to the complexity in the formulation of non-trivial searches various approaches have been proposed to make them more accessible to a wider range of users [20]. Queries become even more complex when the data is stored in a DW historical structure [16]. The objective of research in this area consists in simplifying the information search mechanisms on Historical DW; the strategy consists in making use of a graphical environment without considering implementation details.

5.1 Motivating example

The following query “To determine the quantity of products sold by state and by month”, can be solved automatically by applying our approach. The user just place marks on a diagram, where nodes represent (each with a different icon) the levels of hierarchy as well as the involved measures.

This approach greatly simplifies the task of the user; she/he just should focus on the analysis of the data involved in the query while disregarding the implementation details. Figure 2 displays the graph where the node M-cantidad expresses the measure and the nodes N-producto, N-provincia and N-mes (marked in that order) specify the hierarchy levels used in the query. The temporal queries are performed in the same way, by marking the temporal nodes on the graphic. In both cases, our tool automatically constructs a SQL statement from the marked model. This statement can be executed on a RDBMS.

5.2. Query Graph

The QG allows the HDW users to formulate queries on the storage structure which, subsequently, will be automatically translated to SQL statements. The QG aims to simplify the visualization of the HDW, with the purpose that the users can perform their queries automatically, just by putting marks in the graph and supplying actual values to the parameters. The QG is derived from the HDW and the set of possible queries is established by identifying common query patterns. The QG (Figure 3) consists of a root node that represents the main fact and a set of connected nodes, each of which represents different levels of hierarchies, measures, temporal attributes, temporal entities and temporal interrelation.
6. Informal transformations

In this section we describe the informal transformations that allow us to obtain the QG and the SQL queries derived from the graph.

6.1 Transformation from the HDW to the QG

Below we describe the transformation from the WHD (Figure 1) to the QG (Figure 3). The root node represents the fact in the HDW model, the root name consists of the prefix “H-” attached to the name of the fact. All measures of the fact are transformed to nodes that are next to the root; the node name consists of the prefix “M-” followed by the measure name. The leaf levels in the hierarchy are transformed to nodes attached to the root; the name of the node consists of the prefix “N-” concatenated with the level name. The levels in the hierarchy are transformed to nodes that are connected to the respective leaf levels, the name of the node consists of the prefix “N-” concatenated with the level name. The levels of temporal hierarchies (entities, attributes and interrelationships) are transformed to nodes that are linked to the corresponding nodes; node name consists of the prefix “AT - ET-, RT-” followed by the name of the temporal attribute, entity or relationship, respectively.

6.2 Transformation from QG to SQL sentences

The most frequent and meaningful queries that can be carried out (automatically) on the HDB that is part of the HDW, have into account the lifespan of entities and the valid time of attributes and interrelations. However, other kinds of queries can also be carried out (manually) on the data base. For example, we can obtain the lifespan of temporary entities by formulation a query, i.e. all the temporal ranges in which that entity was active in the universe of discourse. We can establish a pattern for querying temporal entities on the QG (Figure 3), where the nodo-hjid represents the ID of the node associated with the temporal node; E-temp-hjidp is the partial identifier of the temporal node that references the associated node; nodo-hjatr is the descriptive attribute of the associated node, and TI and TF represent the upper and lower bounds of the time interval [TI,TF]. Notice that the identifier of the node associated with the temporal entity is the only variable in the query; consequently, just marking the temporal node (representing the temporal entity) in the QG and supplying the value of the ID of the associated node as a parameter, the query can be automatically derived.

PARAMETERS p tipodato;
SELECT nodo-hjatr, TI, TF
FROM nodo-hj, E-temp-hj
WHERE nodo-hj.id = E-temp-hj.idp
AND nodo-hj.id = p
ORDER BY nodo-hjatr, TI, TF;

Typical queries in a DW (roll-up, drill-down and slice and dice) will be resolved on the HDW by marking in the QG the nodes corresponding to necessary hierarchy levels in a particular order and the measures that are considered as the parameters of the grouping function.

For example, if the query links two hierarchy levels and an aggregate function, we should mark the hierarchy levels in the order that is determined by the primary and secondary grouping criteria and also the measure. The query can be derived and expressed in SQL statements, where hechoid10 and hechoid20 represent partial keys that reference, respectively, to levels nodo-1,0id and nodo-2,0id, respectively, to identifiers of the leaf level. The SQL statement that will resolve this query is the following.

SELECT nodo-1,0id, nodo-2,0id fun(medida1)
FROM hecho, nodo-1,0, nodo-2,0
WHERE hechoid10 = nodo-1,0id
AND hechoid20 = nodo-2,0id
7. Formal transformations

For the specification of the transformation rules is necessary to count with the formal definition of both the source and the target modeling languages. This specification is given through metamodels. In the following section we describe the multidimensional metamodel (Figure 4) and the QG metamodel (Figure 5).

7.1. Metamodels

A metamodel is a mechanism that allows us to define modeling languages, formally. Therefore, a metamodel of a language (either graphical or textual) is a precise definition of its elements given through concepts and rules of certain higher level language (metalanguage). The Meta Object Facility (MOF) [14] defines an abstract and common language for defining modeling languages and how to access and exchange models expressed in such languages.

In our proposal we create domain specific models (DSL) using a focused language that is specific for each domain; in particular, for representing temporal multidimensional models we do not use CWM [5] but a simpler metamodel, instance of MOF. In addition, we design a specific metamodel for the description of the query graph.

Next we present the metamodels used in the transformations:

The multidimensional temporal metamodel (Figura 4) says that the Schema is made of a Fact. A Fact is connected to one or more Dimension and these to cero or more Hierarchy; those are classified in temporal (TempHierarchy) or strict (StrictHierarchy). Hierarchies can be connected to each other. A TempHierarchy is composed by zero or two StrictHierarchy. The Fact, Dimension and Hierarchy are generalized by a multidimensional modeling element (MultidimModelElement) that is formed by Attributes, Identifiers and References. A Fact has zero or more Measures.

The metamodel of the query graph (Figure 5) specifies that the Schema is composed by Vertex that can be a Measure or a Node, the later represents a table in the relational model. A Node is composed by an Attribute, a Key and a ForeignKey. A Node can be a Fact, that represents the main fact of the temporal DW, a Level, that represents the different levels in the hierarchy and Temporal, that represents the temporal nodes; Temporal nodes can be entities (TempEnt), attributes (TempAtt) o interrelations (TempRel); the meta attribute order of the meta class Vertex allows the user to establish the order in the selection of hierarchy levels in the construction of a query.

![Figura 4. Metamodelo Multidimensional Temporal](image-url)
Figura 5. Metamodel of the Query Graph

7.2. Transformation Language

Model transformations of is the main component in MDD. For the transformations from model to model we use ATL [1] that is an 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 MDD development process by supporting the source code generation and other kinds of text generation from the models, such as documentation generation... 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 [19].

For space limitations, the ATL and MOFScript code of the transformations, both from HDW to QG, and from QG to SQL query statements, is not included in this paper, but can be downloaded from [23]

8. Implementation prototype

Below we present some interfaces provided by the prototype and briefly explain its functionality.

Using this tool, the query graph is automatically generated from the temporal multidimensional model (Figure 6).

Once the QG was generated, the next step consists in the marking, which is performed by the application user in order to carry out specific queries. Once done, the SQL statements that resolve those queries are automatically derived by the tool (Figure 7). We have implemented the automatic transformation from the marked graph to SQL text through MOFScript.
Figure 6. Query Graph generation.

Figure 7. Generated SQL code.
9. Conclusion and future work

This work provides a mechanism that allows the end-user to obtain information from a Historical DW without requiring technical knowledge about the structure of the DW. It represents an advance in the field due to the fact that the graphical environment is automatically derived from the Historical DW and then, the SQL statements are also automatically generated from the marked models. The query mechanism is completely implementation-independent. The tool supports typical queries on a DW as well as more complex queries on a HDW. This proposal was realized by a prototype implemented as an Eclipse plug-in that can be downloaded from [23].

The presented work gives rise to potential research lines. Next we detail the topics that are not covered by the present proposal and whose solution requires further development:

1) the development of an efficient query interface: the graphical interface that was presented in the design method and subsequently implemented by the Eclipse prototype is, without doubt, naïve. The efficient use of graphical interfaces involves many considerations such as the type of icons to be used, the different colors, the distribution of icons on the screen, etc. Those features have not been considered in the current work and should be addressed in further iterations;

2) integration of the transformation tool with a DBMS’s: the presented prototype culminates its transformation process with the generation of SQL statements, for the creation of the HDW storage structures on the one hand, and for resolving queries on the HDW on the other hand. Both kinds of SQL statements are not executed directly, but rather the generated text should be exported later to a DBMS. Therefore, another research line to be followed consists in the direct integration of our transformation tool with a DBMS.

3) the use of an object relational data base to build the Platform Specific Model: we used the Relational Model, in particular the SQL92 standard for the development of the PSM. The ORDB databases (standard SQL2003) provide constructions such as user-defined abstract data types that allow a simpler representation of the temporal model. The application of this technology to the development of the PSM is another line of future research.

10. Related work

In this section we introduce the most significant research works related to the graphical query design.

In [3] a prototype for the visualization of query execution in an Oracle DB is developed. Authors claim that the underlying process of running queries in a database is essential to clearly understand the functionality of a DBMS. The aim of that work is to allow students to interactively examine and understand the query process.

In [21] a multidimensional conceptual model is described as a constellation of facts and dimensions composed by multi-hierarchies; they propose a user-oriented algebra for complex queries, and a graphical language based on such algebra to facilitate multidimensional query specification.

In [13] a framework for modeling complex dimensional hierarchies is presented, together with a transformation to a schema-based navigation structure for an OLAP visual interface.

[8] presents an integral interface for queries and visualization of result sets for search and discovery of temporal patterns. Authors propose a temporal query visual language and a navigable visualization of temporal data sets. In addition, the graphical tool facilitates to the user the creation of query patterns.

In [2] a query system based on icons is introduced. This system facilitates the interaction of novice users with a relational database. The goal is to help non-experts to learn and understand the relational data model and a textual query language such as SQL, by using an iconic metaphor.

In [20] a system that allows the user to define software visualizations in a quick and effective way is described. The system uses a visual query language on a variety of data sources to allow the user to specify what information is relevant for the understanding of the tasks and for the correlation of the information.

[17] presents a query language called CQL (Conceptual Query Language) that allows the conceptual abstraction of database queries. The proposal makes use of the expressiveness of the semantic data models to facilitate the formulation of relational database queries.

In [22] a graphical interface called Polaris to explore large multidimensional data bases is presented. As a distinctive feature, the proposal includes an interface for building of visual specifications and the ability to generate a set of relational queries from those specifications.

In [10] a set of visual constructs that allow the user to construct queries in a modular format based on a temporal extension of an ER model is introduced. Related work can be classified into: those works on multidimensional data base queries; those related to temporal data base queries languages; those related to database queries; finally, those works related to the...
design of visual interface with learning purposes, both for the creation of queries and for the understanding of the underlying process of the DBMS. Within this classification, our proposal is a conjunction of all of them due to the fact that our proposed GUI enables the end-user to perform multidimensional and historical queries automatically. The most significant feature that differentiates our proposal from other ones is the design of a graphical environment, derived automatically from the HDW, which allows the automatic generation of SQL statements to perform multidimensional and historical queries. To our knowledge, there is no other proposal using the MDD approach for the automatic derivation of the graphical query environment and the SQL statements from the models of a temporal multidimensional data structure.

7. References


