Designing Data Warehouses through schema transformation primitives

Verónika Peralta, Adriana Marotta, Raúl Ruggia

Instituto de Computación – Universidad de la República - Uruguay
vperalta@fing.edu.uy, adriana@fing.edu.uy, ruggia@fing.edu.uy

Introduction

Data Warehouses (DW) are databases that store information in order to satisfy decision-making requests. DW features are very different from conventional database ones, so that design strategies developed for the latter are generally not applicable for DW. Existing work in DW design consists mainly of techniques for specific sub models (e.g. star and snowflake) and design patterns for specific domain areas, which lack of generality in the treatment of design techniques.

This work addresses the DW design problem through a schema transformation approach. More concretely, this work proposes primitives[1] that abstract and generalise DW design techniques in order to build DW schema relations from the source schema ones.

Existing knowledge concerning DW design mainly consists in practical techniques and methods. One of the main authors is R. Kimball, and proposes practical design guidelines following a star-schema approach [1], [2]. In [3], Adamson and Venerable also present specific solutions for different target business. In [4], Silverston, Inmon and Graziano present DW models in a pattern-oriented approach, and propose techniques for converting a corporate logical data model into the DW model.

In the present work, the application of the design primitives is complemented with two kinds of guidelines: (i) a set of consistency rules that must be always applied to ensure the consistency of the obtained result, and (ii) some strategies that provide different solutions to typical problems that must be faced during the design of a DW.

Furthermore, this work also presents the connection with a multidimensional (MD) conceptual model. This connection supplies the high level vision about the information requirements and it is used as a specification and starting point for the logical design. This work also addresses code generation issues, both for schema and ETL[2] code.

The primitives as well as strategies and rules are currently being prototyped in a CASE tool. The prototype also includes the conceptual model parsing and the graphical interface to the user assistance modules.

[1] The primitive set is listed in Appendix 1. A more detailed description can be found in [10].
[2] ETL: Extraction, Transformation, Loading.

The DW design approach

The approach we follow distinguishes three main steps: (i) conceptual design, (ii) logical design, and (iii) implementation.

The present work focuses in the logical design step and proposes a transformation-based approach to build a relational DW schema.

In our approach, DW logical design is a process that starts with a source schema and ends with a final schema, corresponding to the DW schema. The DW schema is constructed by the application of primitives to the source schema relations. The process also gives rise to intermediate relations.

The primitives are basic transformations applied to a relational database schema in order to obtain a Relational DW schema. Roughly speaking, they take as input a sub-schema[3] and their output is another sub-schema. The primitives also specify the semantics of the resulting relation through a data transformation operation in a SQL-like language.

Some primitives are grouped into families because in some cases there are several alternatives for solving the same problem, or more than one style of design that can be applied.

It is important to note that primitives do not lead to a specific strategy or methodology. Moreover, their application without well-defined design criteria can lead to undesired results, sometimes having consistency problems. In order to help the designer in this aspect we provide: (i) the definition of DW schema invariants, (ii) strategies for solving typical problems that appear in DW design, and (iii) consistency rules for application of primitives.

In addition, the present work takes as input the results of conceptual design step. These results consist of the conceptual schema itself as well as the correspondences with source databases. We use a multidimensional conceptual model[4] and focus on the conceptual representation of MD concepts.

By using the proposed techniques, the designer starts with the conceptual schema and the correspondences. This information is complemented with logical schema design decisions (like to keep value history, or to force (des)normalisation). All these design decisions are

[3] In this work, we consider a sub-schema as a set of relations that are part of a schema.
[4] We are currently using the CMDM model, described in [9].

applied through the primitives and guidelines obtaining the final DW schema.

In order to generate ETL code, the proposed environment makes use of the primitive application trace. At this point, the designer can add other transformations to ensure semantic consistency of data and quality control over it. For example: eliminate data that satisfy a condition (filter), alter an attribute type, change characters to upper cases, etc.

Figure 1 shows the steps of the design process.

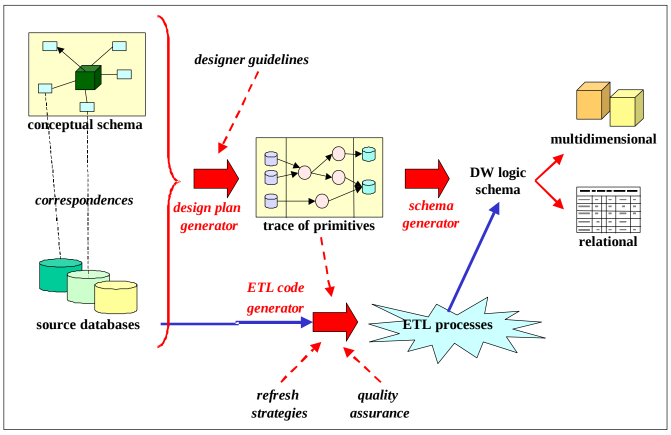

Figure 1 – Design steps.

The design process has three inputs: conceptual schema, source databases and correspondences between them. In a first step the designer indicates some guidelines that complement the conceptual schema. The *design plan generator* assists the designer in determining which primitives should apply. Once the primitives are applied, the *schema generator* builds the DW schema. The primitive trace in addition with refresh strategies, cleansing transformations and quality assurance processes, will be the input to the *ETL code generator*, obtaining the ETL processes. These processes will take data from the source databases and populate the DW.

A CASE tool is currently being implemented to support this design process. The current prototype includes the design primitives (already implemented) as well as graphical interface to the user assistance modules. This prototype is implemented in Java (jdk 1.2) using Jbuilder as the development platform. Parts of the prototype code have been reused from [12] and [13] projects.

The demonstration will consist in the prototype as well as the explanation about the underlying design strategies.

Conclusion

This work presents a DW design approach based on schema transformation primitives. It also includes strategies and rules that assist the designer. These schema primitives enable to design a DW from a source schema acting as design building blocks that have DW design knowledge embedded in their semantics.

Finally, we also present a prototype that implements the set of primitives and their application starting from the conceptual schema.

Acknowledgements.

We would like to acknowledge Alejandro Gutierrez for his contributions and support.

Bibliography

[1] R. Kimball. *The Data Warehouse Toolkit*. J. Wiley & Sons, Inc. 1996.

[2] R. Kimball. *The Data Warehouse Lifecycle Toolkit*. J. Wiley & Sons, Inc. 1998.

[3] C. Adamson, M. Venerable. *Data Warehouse Design Solutions*. J. Wiley & Sons, Inc. 1998.

[4] L. Silverston, W. H. Inmon, K. Graziano. *The Data Model Resource Book*. J. Wiley & Sons, Inc. 1997.

[5] R. Agrawal, A. Gupta, S. Sarawagi. *Modelling Multidimensional Databases*. ICDE' 1997.

[6] L. Cabbibo, R. Torlone. *Querying Multidimensional Databases*. DBPL Workshop 1997.

[7] M. Gyssens, L.V.S. Lakshmanan. *A foundation for multidimensional databases*. Proc. 22nd. VLDB. 1997.

[8] M. Golfarelli, D. Maio, S. Rizzi. *Conceptual design of data warehouses from E/R schemas*. Proc.HICSS-31,VII,Hawaii. 1998.

[9] F. Carpani. *CMDM: Un modelo conceptual para Data Warehouse*. Research Report. UdelaR, Uruguay. 1998.

[10] A. Marotta. *A transformations based approach for designing the Data Warehouse*. Technical Report. UdelaR, Uruguay. 1999.

[11] V. Peralta. *Técnicas de integración de diseño conceptual y lógico de Data Warehouses*. Research Report. UdelaR, Uruguay. 1999.

[12] P. Garbusi, F. Piedrabuena, G. Vazquez. *Design and Implementation of a schema transformation based DW design tool*. Graduate Project, ongoing work. UdelaR, Uruguay.

[13] A. Picerno, M. Fontan. *An editor for CMDM*. Graduate Project, ongoing work. UdelaR, Uruguay.

Appendix 1 – The set of primitives

Primitive		Description
P1	**Identity**	Given a relation, it generates another that is exactly the same as the source one.
P2	**Data Filter**	Given a source relation, it generates another one where only some attributes are preserved. Its goal is to eliminate purely operational attributes.
P3	**Temporalization**	It adds an element of time to the set of attributes of a relation.
P4	**Key Generalisation ***	These primitives generalise the primary key of a dimension relation, so that more than one tuple of each element of the relation can be stored.
P5	**Foreign Key Update**	Through this primitive, a foreign key and its references can be changed in a relation. This is useful when primary keys are modified.
P6	**DD-Adding ***	The primitives of this group add to a relation, an attribute that is derived from others.
P7	**Attribute Adding**	It adds attributes to a dimension relation. It should be useful for maintaining in the same tuple more than one version of an attribute.
P8	**Hierarchy Roll Up**	This primitive does the roll up by one of the attributes of a relation following a hierarchy. Besides, it can generate another hierarchy relation with the corresponding level of detail.
P9	**Aggregate Generation**	Given a measure relation, this primitive generates another measure relation, where data are resumed (or grouped) by a given set of attributes.
P10	**Data Array Creation**	Given a relation that contains a measure attribute and an attribute that represents a pre-determined set of values, this primitive generates a relation with a data array structure.
P11	**Partition by Stability ***	These primitives partition a relation, in order to organise its history data storage. Vertical Partition or Horizontal Partition can be applied, depending on the design criterion used.
P12	**Hierarchy Generation ***	This is a family of primitives that generate hierarchy relations, having as input, relations that include a hierarchy or a part of one.
P13	**Minidimension Break off**	This primitive eliminates a set of attributes from a dimension relation, constructing a new relation with them.
P14	**New Dimension Crossing**	This primitive allows materialising a dimension data crossing in a new relation. It also is useful to simply de-normalise relations, which improves queries performance.