Designing relational data warehouses through schema-transformation primitives

A Prototype[‡]

Alejandro Gutiérrez, Adriana Marotta

Instituto de Computación, Facultad de Ingenieria, Universidad de la República, Montevideo, Uruguay

June 2001

Abstract

The logical design of a Data Warehouse (DW) is a task that requires the application of techniques and strategies that are specific of DW context. In [Mar00] we present a mechanism for designing DWs. Based in this mechanism we developed a prototype of a DW design tool. This tool offers a graphical user interface that allows the designer to apply transformation primitives to a source schema constructing a DW schema, visualise the generated transformation trace, check DW schema invariants and apply consistency rules.

[‡] This work was supported by Comisión Sectorial de Investigación Científica from Universidad de la República, Montevideo, Uruguay

1. Introduction

We have developed a prototype of a DW Design tool, called *DWDesigner*, which can be combined with other components conforming a CASE tool. These other components have been developed in the context of different projects of the CSI group[1]. The components are the following: (1) *CMDM (Conceptual Multidimensional Data Model)* [Pic99], (2) *A Repository Manager for a CASE tool* [Arz99], and (3) *From the conceptual schema to the logic schema of a DW* [Per00].

Figure 5.1 presents an overview of the architecture of the whole CASE environment.

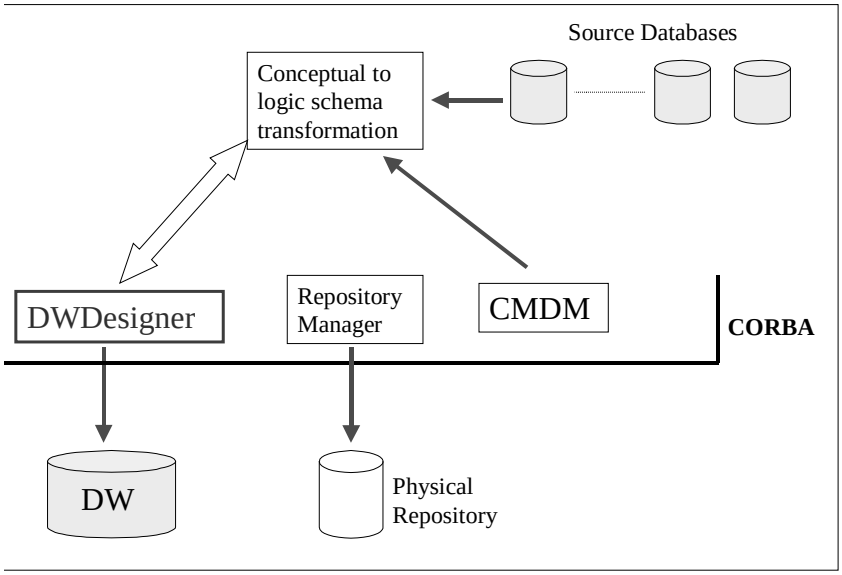

Figure 5.1

DWDesigner is a tool that implements the principal ideas about DW logical design, presented in [Mar00]: transformation primitives, transformation trace, schema invariants and consistency rules.

The prototype is a DW Design tool with the following main characteristics:

- It allows the designer to design a DW schema starting form a source schema by means of Primitive applications.
- It generates a trace of the transformation applied.
- It provides invariants checking.
- It includes consistency rules triggering.

[1] CSI group – Grupo "Concepción de Sistemas de Información". Instituto de Computación. Facultad de Ingenieria.

- It has a graphical user interface.
- It is extensible. Its modular design allows adding and removing primitives without modifying or re-compiling the existing code.

In Section 2 we present a brief description of the prototype and in Section 3 we present the conclusions.

2. Prototype description

In this section we pretend to give a descriptive view of the prototype.

First we present a summarised analysis of the tool's functional features, then we present the most relevant aspects of the conceptual design, and finally we make some comments about the implementation.

2.1. Functional Features

The tool's main features were enumerated in previous section.

The tool is intended to be a DW design graphical environment. It should be useful to a DW designer who has a source database, some DW requirements, and wants to generate a DW schema that satisfies these two elements. With this tool the designer can apply different DW design criteria and techniques in order to obtain the target DW schema.

Invariants checking can be invoked at any moment in the design process. It allows checking the consistency of the schema that is being generated.

Consistency rules are triggered by the application of certain primitives to certain elements; when these applications put in danger the consistency of the schema that is being generated.

The graphical interface facilitates interaction with the source database elements, application of primitives and parameters selection, and visualisation of the design trace.

Figures 5.2, 5.3, 5.4, 5.5 and **5.6** show the interface of the tool.

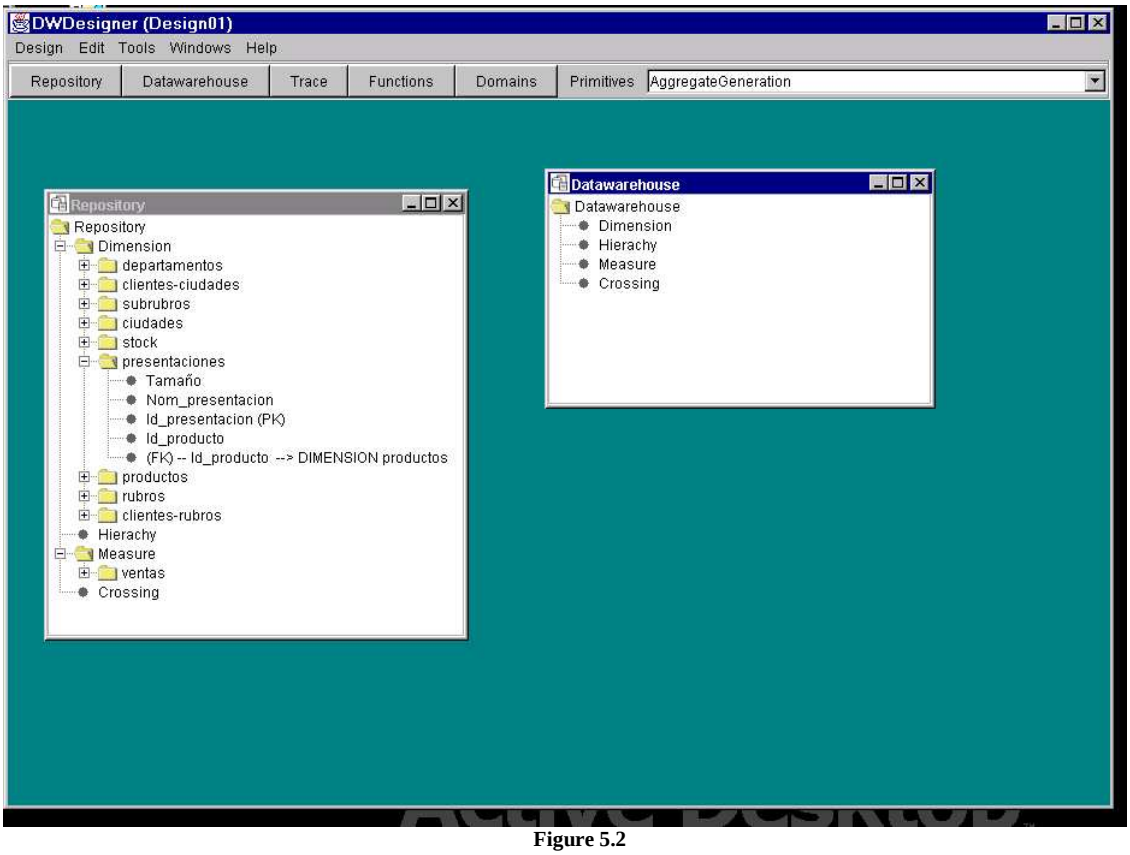

Figure 5.2

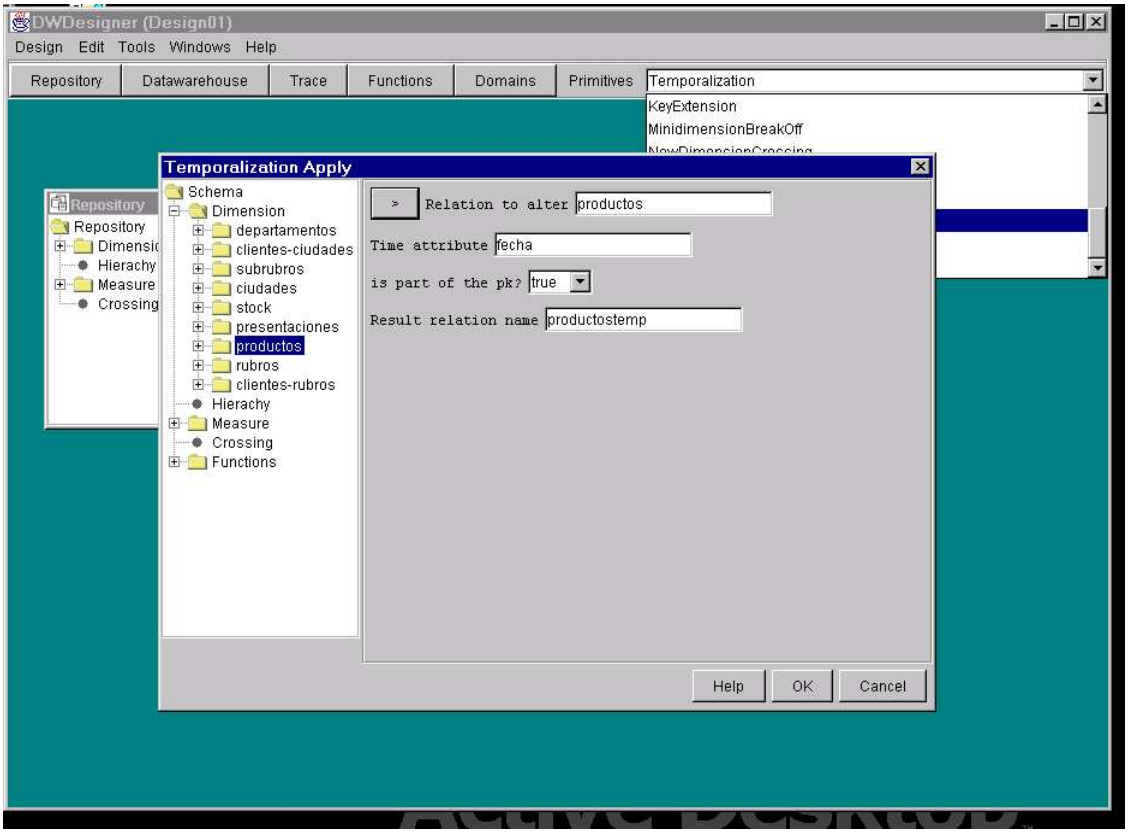

Figure 5.3

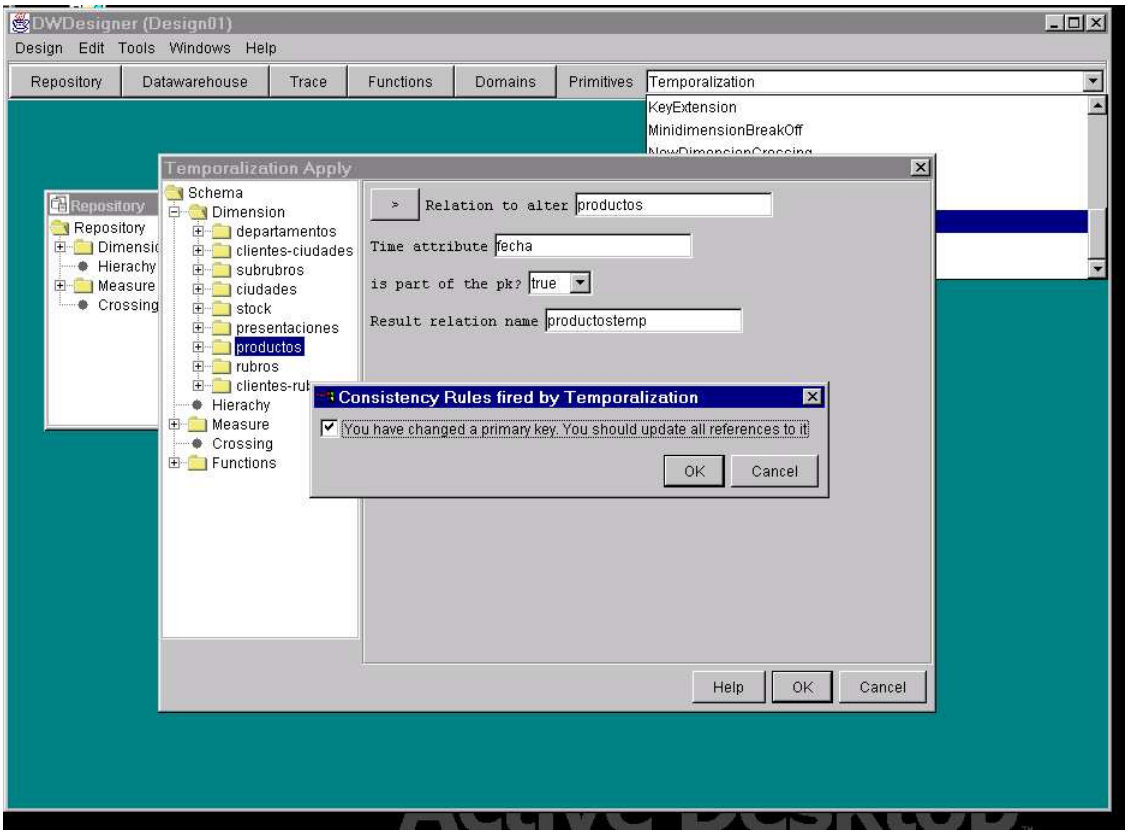

Figure 5.4

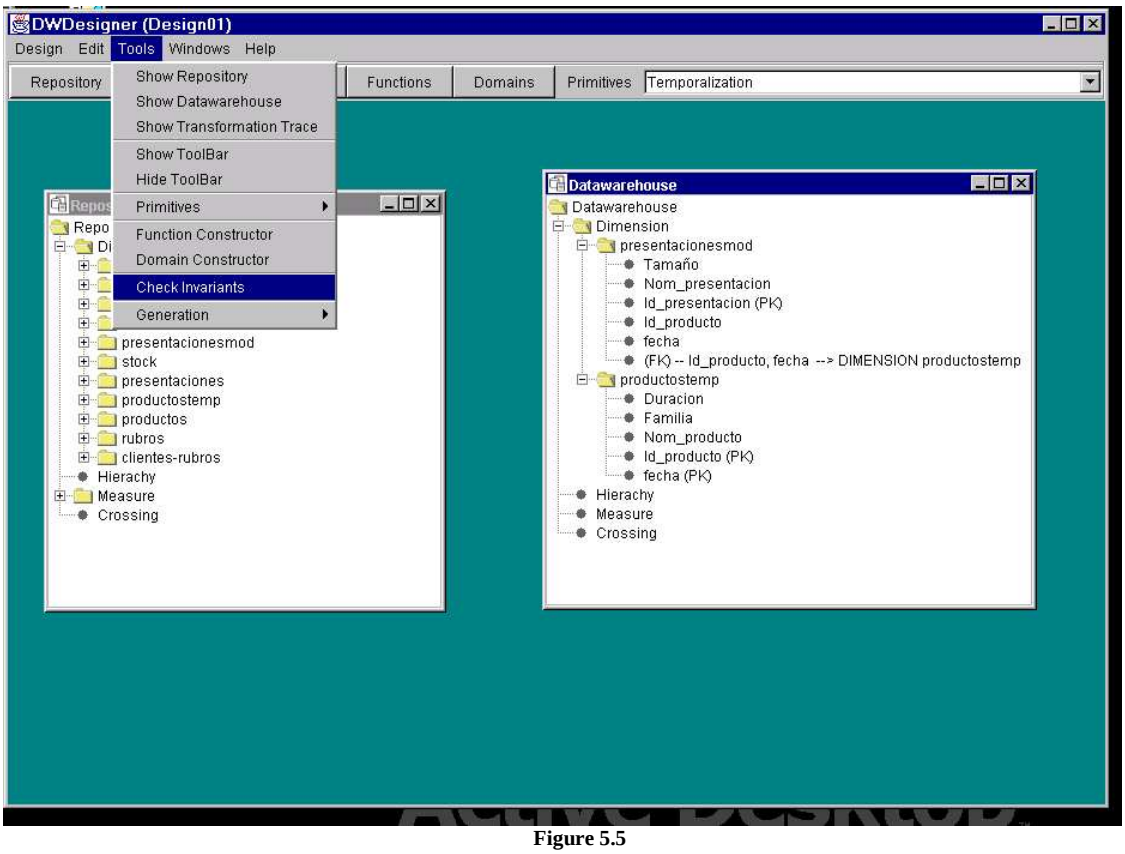

Figure 5.5

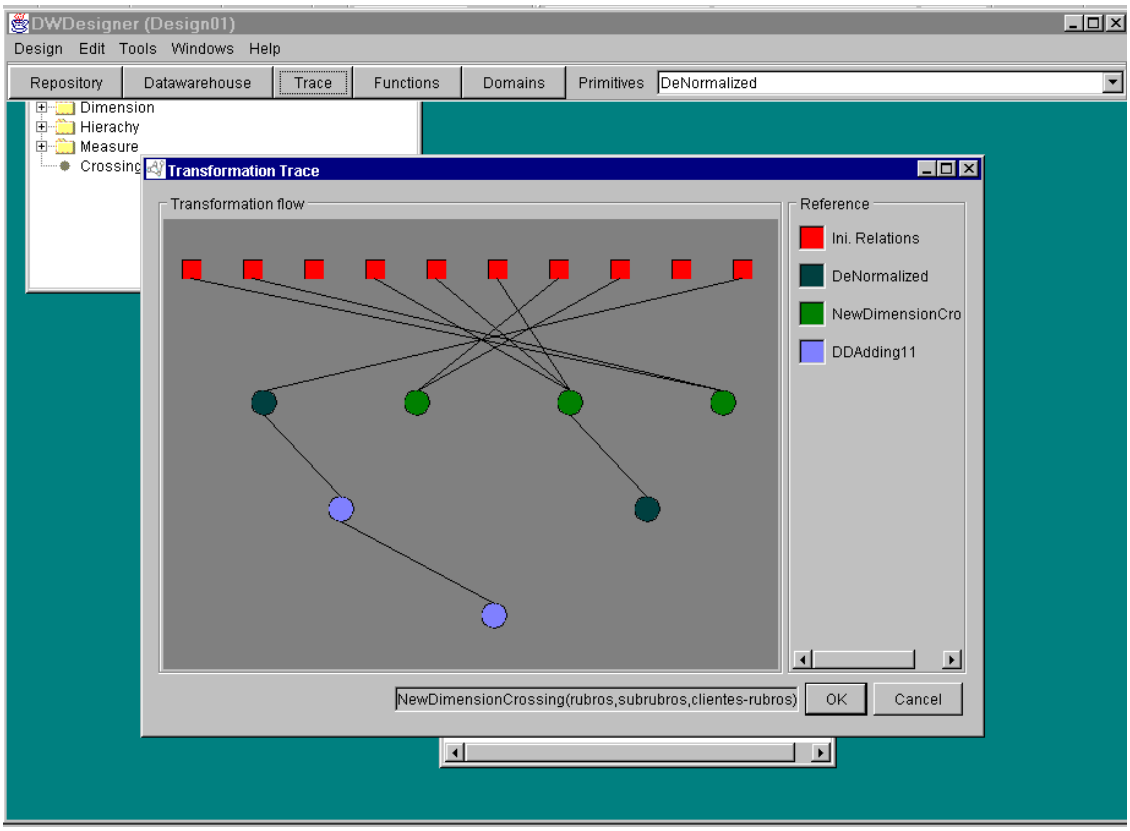

Figure 5.6

2.2. Conceptual design

For the design of the tool we applied object-oriented techniques. The design language we used was UML.

The core of the tool is the *Virtual Machine*. This is the most important layer of the system. The main components of the *Virtual Machine* are the following:

1) Tools for schema transformation

   This is the representation of: the transformation primitives, and two sets of schema elements (called Repository and Datawarehouse) that are the containers used during the transformation process.

2) Database concepts representation

   All schema elements, relations, attributes, keys, etc. must be represented in order to be manipulated by the user and the primitives.

3) Trace

   Each applied primitive, with its input and output relations, is recorded in the trace.

4) Invariants

   The invariants are properties that must be satisfied by the schema. Therefore we represented the invariants as a part of the schema representation.

5) Consistency Rules

   We represented the Rules as an independent entity, which is referenced and invoked from different places.

The class diagram of the DW design tool can be seen in the Appendix.

2.3. Implementation

The prototype was implemented in JAVA language, using the Java Development Kit version 1.2.2 (JDK 1.2.2) and Borland's JBuilder version 2 as the development environment.

Implementation details of the tool, excepting the parts of Invariants and Rules, can be found in [Gar00]. In this Section we briefly explain how we implemented Invariants Checking and Rules Triggering.

Invariants Checking is an option of the tool's main menu that allows checking the consistency of the existing DW schema. The user has the possibility of choosing between a list of invariants. We implemented the procedures that perform the Invariants Checking as methods of the *General Schema* class. When the user press *OK* button, we call the methods of the *General Schema* class that correspond to the invariants selected by the user.

Rules are implemented as an abstract class *Rule* and a sub-class *RuleXX* for each existing rule (analogous to the primitives). We defined the *RuleDirectory*, which is a sequence containing the existing rules and is initialised at start. Also at this moment, some primitives initialise the *rules* attribute that is a set of rules (referencing to rules of the RuleDirectory). The rules that belong to a primitive's set of rules are the ones that must be triggered after the primitive application. When the user applies certain primitives, a dialog box appears showing him what rules should be applied in the form of check boxes. The rules that are checked by the user are applied automatically. The transformations that are made by the rules, are applications of primitives, therefore they have the same behaviour as any primitive application (e.g. they are reflected in the trace).

3. Conclusion

DWDesigner is a prototype of a DW Design Tool.

The developed tool implements the principal ideas of [Mar00]: transformation primitives, transformation trace, schema invariants and consistency rules. This tool offers a graphical user interface that allows the designer to apply primitives to a source schema, constructing a new schema, visualise the generated transformation trace, check schema invariants and apply consistency rules.

The tool can be connected with other modules, complementing each other. Altogether, they constitute a CASE tool for designing a DW that covers the stages of: conceptual modelling, derivation of a logical model, management of the logical model, and persistency of the design.

References

[Arz99] G. Arzua, G. Gil, S. Sharoian. *Manejador de Repositorio para Ambiente CASE*. Facultad de Ingenieria. Universidad de la República del Uruguay. In.Co. Proyecto de Taller 5. 1999.

[Gar00] P. Garbusi, F. Piedrabuena, G. Vazquez. *Diseño e implementación de una herramienta de ayuda en el diseño de un Data Warehouse Relacional.* Facultad de Ingenieria. Universidad de la República del Uruguay. In.Co. Proyecto de Taller 5. 2000.

[Mar00] A. Marotta, Designing relational data warehouses through schema-transformation primitives. Reporte Técnico INCO-01-10. InCo - Pedeciba, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay. Diciembre 2000. ISSN 0797-6410.

[Per00] V. Peralta *Sobre el pasaje del esquema conceptual al esquema lógico de un Data Warehouse.* Facultad de Ingenieria. Universidad de la República del Uruguay. In.Co. Reporte Técnico. 2000.

[Pic99] A. Picerno, M. Fontan. *Un editor para CMDM*. Facultad de Ingenieria. Universidad de la República del Uruguay. In.Co. Proyecto de Taller 5. 1999.