

TEMPTOOL – A Tool For Temporal Data Modelling

Eugênio de Oliveira Simonetto*, Duncan D. Alcoba Ruiz**,
Cristiano Ramos Moreira, and Glauco Valim dos Santos

Universidade Federal do Rio Grande do Sul
Pontifícia Universidade Católica do Rio Grande do Sul

Abstract. This work presents a tool that allows the users responsible for the temporal data of an information systems to get all the needed structures for the representation of temporal aspects in their databases. This tool can be used with most important commercial DBMS.

1 Introduction

In the last years several temporal data models have been proposed. This fact shows us the great importance of the temporal representation in data modelling.

In the conventional relational database systems, the data are stored in two dimension tables, where rows are tuples and the columns are attributes [2]. The introduction data associate with time factors is possible to realise the table with one more dimension, the time [7].

For a temporal database implementation in RDBMS there are two approaches. The first is to extend the relational model semantics for the time aspects incorporation. The other approach is to implement the temporal database on a conventional relational model with the temporal aspects being additional attributes. As a first approach example, we have TRM [4], HSQL [5] e TSQL2 [6].

In the second approach the time aspects are not completely supported by the DBMS. So, to translate the queries and updates involving time to the data manipulation language is a task of the system designer.

The relational DBMS time aspects incorporation, according the second approach, could be done by the additional attributes insertion that represent the transaction or valid time begins and (or) ends [6]. The [1] model proposed incorporates begin and end transaction time attributes. The model also does not extend the DBMS semantics for the temporal information handle. It just uses the DBMS support to handle with this information.

* eosimonetto@ea.ufrgs.br, Universidade Federal do Rio Grande do Sul, PPGA, Porto Alegre – RS, Brasil

** duncan@inf.pucrs.br, Pontifícia Universidade Católica do Rio Grande do Sul, FACIN, Porto Alegre – RS, Brasil

2 The [1] Temporal Model

In the model developed, we considered that the database designer could not directly handle with the temporal data. This means that the conventional user can still visualise (and handle with) the database as being a conventional database. Taking into account the possible ways of temporal representation in databases, it implies that, in the proposed model, temporal aspects should be represented by transaction time. This is the only way that permits the time line used linked to the temporal data, need not be handle by the user, since he/she makes use of the DBMS time.

For the representation of data temporal aspects, we adopted the database state concept [5], where each state is defined by its attribute values. Any attribute value update generates a new state in the database. Each state also has its determined duration that is represented by a time interval in which the attribute values remains unchanged. Consequently, each database instance is labelled with a temporal interval.

The temporal representation type, adopted by this tool was time interval. Therefore, to represent the transaction time, two attributes are added: start of transaction period (S_TT) and end of transaction period (E_TT).

In temporal databases, update operations should be handle differently from the conventional database updates, to assure that any data, that becomes older, must not be lost. Consequently, every update should be followed by operations over the older value for its maintenance. For such updates it has been developed routines responsible for the update of transaction time automatically. These routines are not visible by the conventional database user.

The application designer defines time granularity in the model. For validation purpose, seconds granularity was used.

3 TempTool – The Proposed Tool

In temporal database, update operations should assure that older data must not be lost. Therefore, every update should be followed by operations on the older value for its maintenance. In the proposed model [Sim00], for such updates was developed routines (making use of DBMS triggers concepts) responsible for transaction time automatic update in the modelled tables. Manual construction of this structures are hard and exhaustive, since the large number of code lines, including triggers and tables to be produced.

To handle with these difficulties, it has been developed a support tool for this development phase, according with the [Sim00] model. The tool use starts after the end of project implementation, making use of a CASE tool, and before loading specifications in the DBMS. DB-MAIN [3] is a CASE tool that operates in this way: at the project end, the database schema generation is done into a text-type file called database script. After that, the database administrator loads DBMS with the script. To make changes in such scripts it is not necessary to modify designer teams' routines. Just adding one more step for temporal aspects incorporation in the database application.

The module responsible for the SQL script recognition and creation of new tables and routines is ready to use. Our next step is the development of a graphical module of the tool that allows the designer to develop his/her conceptual model directly in the tool.

3.1 Tool Specification

For the specification of the data model development support tool, in accordance with [1] temporal model, we have worked to identify the several stages of the model development process, and stages that could become supported by any kind of computational tool. The following stages was identified:

1. Initial schema analysis;
2. Attribute classification;
3. Model structures creation.

After these three stages, the final script is built, in accordance to the proposed model, and it is ready to be submitted to a DBMS. This script is composed by all the triggers and tables derived from the initial database schema. The only stage with the user interference is the second one, attribute classification, where the attributes must be classified as temporal or static, conforming to the needs of database designer.

Initial Schema Analysis This first stage involves the database recognition to be submitted to the tool. In this stage all the database component tables will be identified, and every attribute of each table, including primary keys as well. This stage has the objective to prepare the attributes to be temporally classified in the next stage.

The database schema to be recognised by the tool should be supplied as a script text file generated by a CASE tool or even for the application designer.

After this first stage, where the structures submitted to TempTool are recognised, starts the second stage.

Attributes Classification In this stage, the application designer should identify each table attribute as static or temporal according with the proposed model. It is the stage where the designer user plays with the tool behaviour.

A table, besides it's primary key attributes, will be able to have none or some temporal attributes, and none or some static attributes.

In the attributes classification interface (see figure 3.1), the tool supplies each table attributes to be classified by the application designer. After classification, it is presented the **Script** option, and this option will produce the new script, according to the Simonetto [1] temporal model. This stage also asks the user about for what kind of DBMS should the script to be generated, since triggers definition and syntax could present some differences among distinct DBMS. After the attribute classification by the application designer, it will start the structure creation, as the model description.

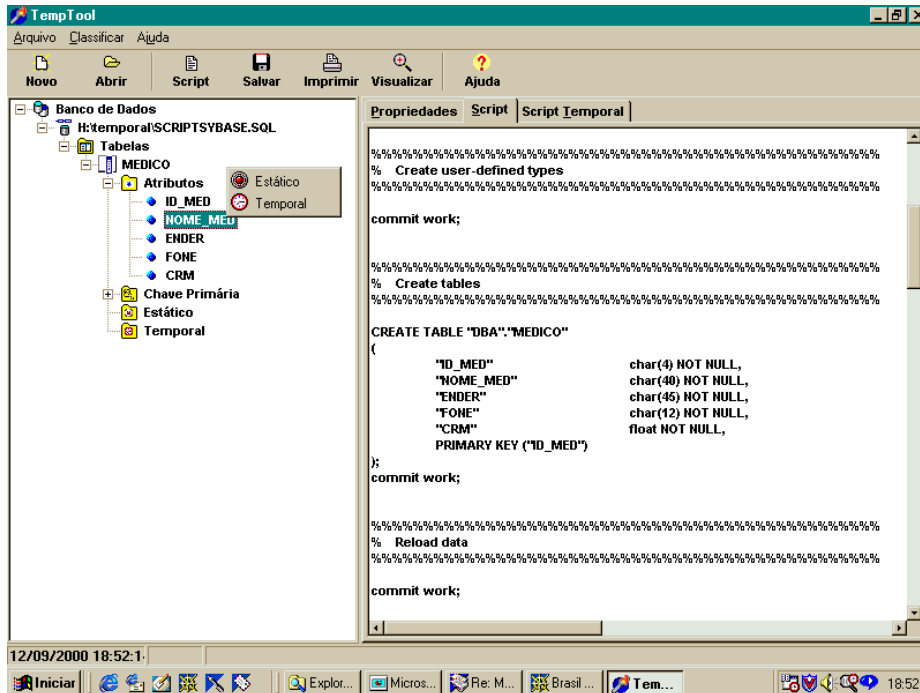


Figure 3.1 – Attributes Classification Interface

Creation of Model Structures In this last stage of tool functionalities, it is necessary to be produced the structures that will assure the model database maintenance and temporal data automatic management.

According with the attribute classification, executed in the previous stage, the component structures are determined. These new structures imply in triggers and procedures creation and new tables, into a script file.

The first structure to be generated by the tool is the original user definition, called by [1] model *instantaneous table*. For each attribute classified as temporal, it is created a *new table* and *temporal data maintenance routines* to handle with this kind of data at every insert, update and delete on *instantaneous table*.

One only table is defined for all attributes classified as static (they will make part of the same table) as their respective *temporal data maintenance routines*, in case of the attribute insertion and deletion (since past values of static attributes need not be kept). The way the new script is presented by the tool is showed in figure 3.2.

4 Concluding Remarks

This work presented a tool for temporal databases development and maintenance, called TempTool. TempTool allows the information system designer to

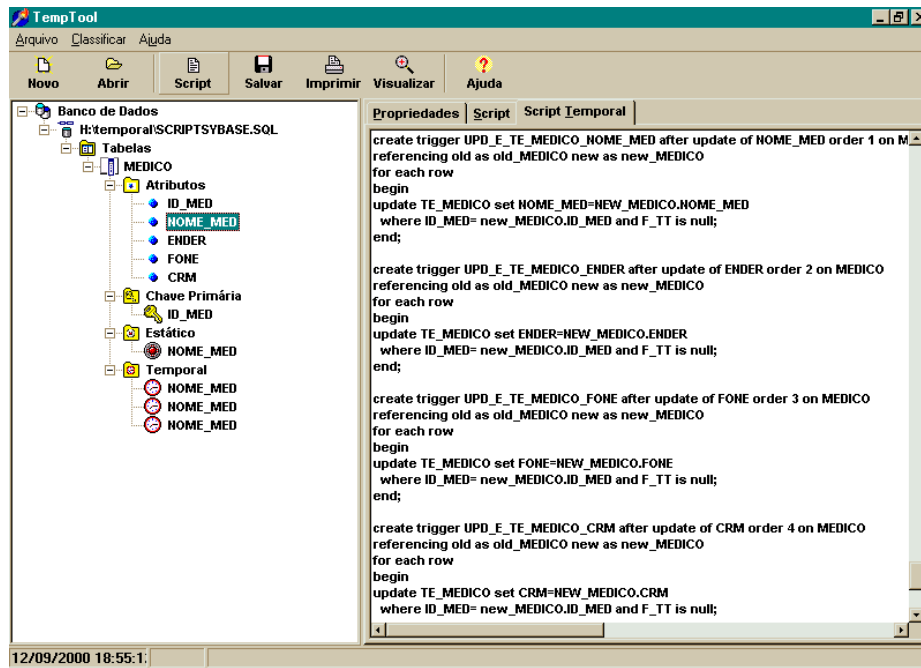


Figure 3.2 – Temporal Script Presentation Interface

use resources to generate, after the development of the database implementation design, and before DBMS load (as Oracle, Sybase and Ingres), a new data model with the necessary structures for the storage and maintenance of data which past values are important to be retained into the DB.

Usability tool results was consider excellent, since the tool correctly “translate” the original scripts of the user data models into temporal data models.

5 Future Work

In the first tool development stage for validation purpose, it was implemented the recognising and script generation for Sybase, Oracle and Ingres DBMS. Our next step in the tool development, is the script generation for Informix and SQL Sever DBMS. After that, will be developed a way to allow the user to design his/her E-R like diagram in TempTool. Thus, the attribute classification could be done in the system conceptual modelling, for subsequent generation of final scripts for temporal databases.

References

1. Simonetto, E.O.; Ruiz, D.D. *A proposal model to incorporate temporal aspects to the relational dbms using active databases concept*. In: DATABASES & INFORMA-

- TION SYSTEMS. 4TH IEEE INTERNATIONAL BALTIC WORKSHOP., 2000, Vilnius. Proceedings of the 4th Ieee International Baltic Workshop. Vilnius, Lithuania: VGTU, Lithuanian Computer Society (ed. Albertas Caplinskas), 2000. p.52-58.
2. Codd, E.F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, v.13, n.6, 06/1970.
 3. DB-MAIN Case. Db-Main Tutorial. <http://www.info.fundp.ac.be/~dbm>.
 4. Navathe, S.; Ahmed, R. Temporal Extensions to Relational Model and SQL. In: *Temporal Databases: Theory, Design and Implementation*. Benjamin /Cummings, 1993.
 5. Sarda, N.L. A Historical Query Language. In: *Temporal Databases: Theory, Design and Implementation*. Benjamin/Cummings, 1993.
 6. Snodgrass, R.T. The TSQL2 Temporal Query Language. Boston: KluwerAcademic Publishers , 1995.
 7. Tansel, A.U. Modelling Temporal Data . *Information and Software Technology*. V. 32, n. 8, p.514-520, 1990.