Vahid RAFE[2], Shahram JAMALI[3], Mohsen RAHMANI[2], Farzaneh MAHDIAN[1], ,

Islamic Azad University, Tuyserkan Branch (1), Arak University (2), University of Mohaghegh Ardabili (3)

From Class Diagrams to Relational Tables: A Graph Transformation-based Approach

Abstract. *Nowadays UML has been accepted in both academia and industry as a de facto modelling language to develop software systems. As relational databases are widely used for developing software systems, thus, it is a challenge to derive relational tables from designed models. In this paper, we present a formal yet automatic approach to extract relational tables from class diagrams. To do so, we adopt graph transformation systems. We have designed some graph transformation rules to derive necessary tables from UML class diagrams. All the necessary concepts are illustrated through a case study.*

Streszczenie. *Język UML jest powszechnie akceptowany jako język do tworzenia oprogramowania. Jako baza relacyjna jest powszechnie wykorzystywany chociaż nie jest łatwo tworzyć na jego podstawie tablice relacyjne. W artykule przedstawiono automatyczną metodę ekstrakcji takich tabel. W tym celu zaadaptowano transformację grafu. (**Od funkcji klasy do tablic relacyjnych: metoda wykorzystująca transformację grafu**)*

Keywords: graph transformation systems, class diagram, and verification.
Słowa kluczowe: transformacja grafu, tablice relacyjne.

Introduction

Although UML [1-2] has been accepted in both academia and industry as a de facto visual modeling language, it is, however, non-formal and inaccurate. There is no complete method to check it. Additionally, relational database and databank design is one of the most important parts in producing software in software engineering sciences. Simplicity, accessibility, powerful theoretical support, powerful software tools, security, integrity, normalization, and optimization of query are among advantages of the relational model.

Uncertainty on the accuracy of the software performance causes difficulties in producing software in traditional methods. Application of formal methods to produce software will reduce this uncertainty. The application of formal methods using mathematical abstractions is particularly useful in the field of software description. They can remove the gap in software production process and enhance the capacity of the project. So, in case of availability of automatic tools for generating database tables and automatic executable code so that they can be approved and checked, we should be able to produce software with higher confidence and guarantee more reliability.

UML diagrams and relational tables are both models. In most cases, to express model transformations, we can use graph transformation. What's more, the transformation by visual models can be formulated thorough graph transformations, because they are suitable for describing model basic structures.

During transformation of the model via graph transformation, source and destination models are given as graph. Transformation by the use of graph transformation means that the graph takes the abstract syntax of a model to transform it according to transformation rules. The result of this transformation shall be the creation of destination model. To do this, there are various models. In this paper, we have selected AGG tool.

AGG is an environment to develop attributed graph transformation system supporting algebraic approach. AGG suggests typical graph transformations including inheritance and multiplicity. It also controls the existence of compatibility between graphs and graph transformation system. It can find any conflict between rules by performing critical pair analysis. Hence, it is an appropriate tool for us to implement our goals.

The purpose of this paper is to employ algebraic graph transformation system on UML model to extract relational database tables and, finally, to analyze the system in terms of syntax reach ability and verification. We considered the use of AGG as a tool for modeling graph transformation.

The rest of the paper is organized as follows: Section 2 surveys state of the art approaches for extracting relational tables. Section 3 shows the proposed approach to extract relational tables from UML class diagrams and Section 4 concludes the paper.

Related works

Graph transformation has been taken into consideration in both industrial and scientific fields. Researchers have already been conducted over different issues in this regard. Below is a short description of the papers in which one of the graph transformation tools were used to transform UML model to relational database model:

Michael Lawley, in [3], Introduced DATC [3] model transformation tool and TefKat [3] language, the authors explained core concepts of transformation rules, relationships evaluation and models definition. They examined the mapping of an object-oriented diagram as source model, and a relational database as destination model. They also studied the way in which the employed language to support the mapping ability and the reuse of transformation characters which are required to have a successful MDA [3] as an approach to generate high scale and long-life systems.

Jorn Bettin [4] presented concepts for a real syntax in model to model transformations, which are useful in industrialization of commercial software. In his paper, Jorn Bettin studied the transformation of UML [2] model to RDBMS model as an example of model to model transformation in which model's elements are inscribed form independent PIM platform to special PSM platform. The addition of transformation characters using OCL [4] in destination model elements and a textual model language for the mapping of source model elements and their transformation to destination model elements, as well as a visual description of model to model transformation by showing meta-models from the start to the end and by symbolization.

Gergely Varro' et al. [5-7] introduced a new approach to implement a graph transformation engine based on RDBMSs (Relational Data Base Management System). The nature of this approach is to generate a database for each

rule and for using pattern adaptation via internal linkage operations. As a result, the authors of this paper obtained a powerful and fast transformation motor which is particularly suitable for followings: 1) Model development tools with a systematic RDBMS storage, and 2) embedded of model transformations inside large distributed applications in which models have been emphasized in a relational database as a recurring basis. It is also essential to control compatible large models.

Our proposed approach

We know that UML [1-2] class diagrams are composed of stable and unstable classes, initial attributes, dependences, varieties, and inheritance classes. On the other hand, relational database comprises tables with a record in each row and a field in each column. One or several columns here are considered as primary key, and the columns relating to tables are called foreign key.

By comparing these two models, we will find out that the mapping of their components is possible. In general, performing following operations is necessary [8]:

1. Calculation of various closures from class inheritance;
2. Collection of attributes and dependencies from sub-classes;
3. Collection of attributes and dependencies from unstable connected classes;
4. Generation of tables;
5. Generation of columns;
6. Removal of help structures;
7. Removal of source model.

The above-mentioned operations can be done according to following basic rules [9]:

1. Mapping stable classes to tables.
2. Mapping attributes of dependency from a stable class to the relevant column.
3. Mapping dependencies and preliminary attributes of unstable classes to the columns generated in the tables out of reference stable class.
4. Mapping main class attribute inheriting form another class to a column equivalent to the main class.

In this section, we study the mapping of UML class diagrams into relational database tables using a standard method. The transformation (with various modifications) has been used a lot as a base for transforming high level applicative relational models.

The purpose of this paper is to automatically extract relational tables out of UML class diagrams by using graph transformation systems.

However, to transform class diagrams to a relational model, the aforementioned diagrams will need some modification: there is no inheritance relationship in relational tables. Thus, we need to transform inheritance links between classes into ordinary links. To this end, we should transfer all the links of child classes such as attributes and dependencies to the biggest supper class.

Determining the characteristics of UML class diagrams and relational models:

Here, we determine the characteristics needed to generate UML class diagrams and relational models, and to transform the model. These characteristics have been specified for implementing the concepts of UML class diagrams and relational models in the graph [10].

UML class diagrams: include regulated classes in an inheritance hierarchy (with parent characters). Classes have attributes (attrs) which are categorized according to classes or primitive data types (Primitive DT). Dependencies conduct a direct relationship from a source class (src) to a destination class (dst). In Fig. 1, there are some notations of class diagram in UML.

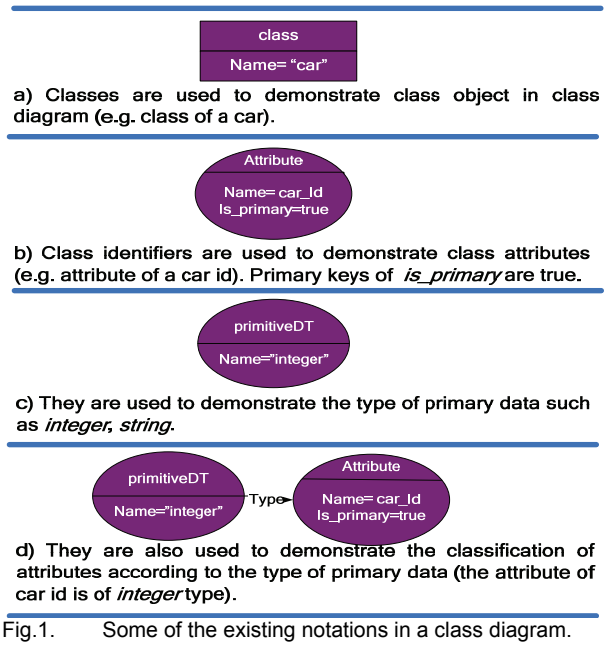

Fig.1. Some of the existing notations in a class diagram.

Relational database: it includes tables composed of columns (tcols). Each table has a column which is a primary key (pkey). Foreign key limitations have been attributed to the tables (fkeys). This limitation is referred to main columns (cref) of source and destination tables (tref). It also corresponds to the columns of dependency tables (kcols). In Fig. 2, there are some notations of relational tables.

Rules of graph transformation: Now, we deal with some rules used in the proposed model. We begin model transformation with transitive of the inheritance relation. This transitive creates a direct relation through parent edge in between the classes where there is medium inheritance relation. This step is formulated in Fig.3.a by the rule of parentClosureR. This picture shows the left and the right side of the rule. The mapping between the two sides of the rule is shown with similar numbers.

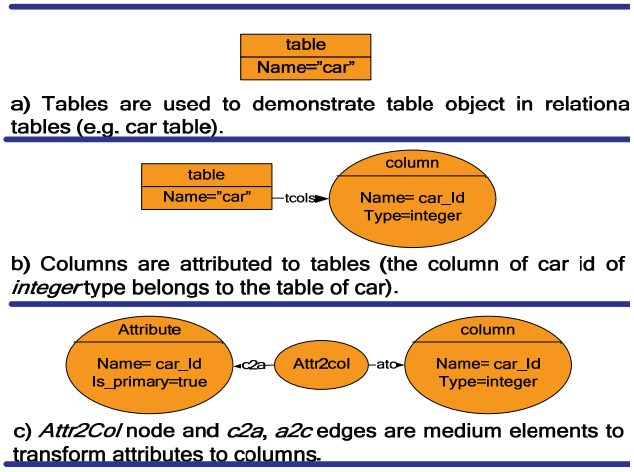

Fig.2. Some of the existing notations for relational tables.

There is a negative applicative condition (NAC), is shown in Fig.3.b, which is called noParent3/1 and prevents regeneration of parent. Before applying other rules, it is necessary to determine completely the inheritance relations

between classes by the rule of parentClosureR. This rule is therefore placed in the first layer, i.e. layer 0.

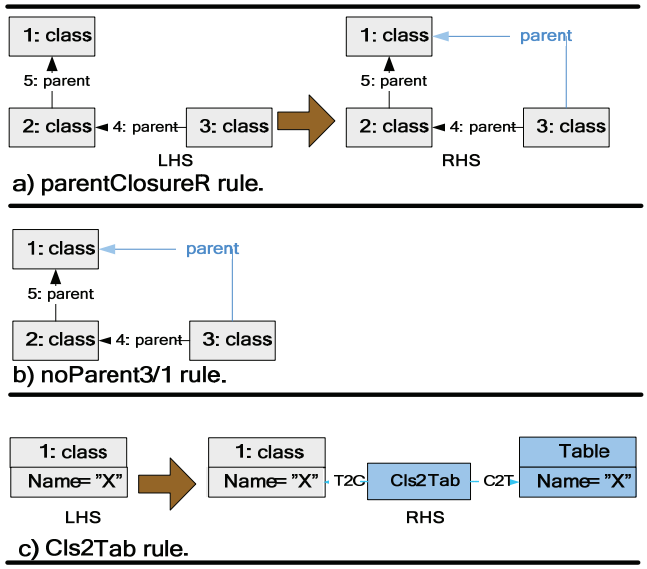

Fig.3. Some of the proposed rules.

We continue model transformation by mapping classes to the tables. In this stage, a table with a similar name is created for each class. The relation between these two is guaranteed by medium edges of t2c, c2t and Cls2Tab. This application is shown in Fig.3.c via the rule of Cls2Tab.

There are negative application conditions or noCls2Tab which limits the application of the rule in the equal mapping for one time.

Conclusion

In this paper, we presented a formal framework to extract relational tables from UML class diagrams using graph transformation systems.

To do so, we designed different rules to transform classes to tables, attributes to columns, dependency to table and foreign limitations, and to determine specifications of the elements of the destination model.

Class diagram taken into account in this approach as source model was a normal class diagram. In the future, this approach can be extended by designing normalization process to accept a wider range of class diagrams as source model.

REFERENCES

[1] Y. Ou, "On Mapping between UML and Entity-Relationship Model," in *intrnatinal workshop,uml'98'* mulhouse, pp. 45-57, june, 1998.

[2] C. Anderton, L. Barroca, D. Bowers and M. Newton, "Converting Class Diagrams to Object-Relational Schema: Using the MDA to exploit the expressive power of the full SQL," 1744-1986 30th, April, 2009.

[3] M. Lawley, K. Duddy, A. Gerber and K. Raymond, "Language Features for Re-Use and Maintainability of MDA Transformations," *In OOPSLA workshop on Best Practices for Model-Driven Software Development,* Vancouver, Canada, October, 2004.

[4] J. Bettin, "Ideas for a Concrete Visual Syntax for Model-to-Model Transformations," *OOPSLA Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003.

[5] G. Varro', K. Friedl and D. Varro', "Implementing a Graph Transformation Engine in Relational Databases," *Software and Systems Modeling,* vol. 5, pp. 313-341, September, 2006

[6] G. Varro' and D. Varro', "Graph Transformation with Incremental Updates," *Electronic Notes in Theoretical Computer Science,* vol. 109, pp. 71-83 14, December, 2004

[7] G. Varro', K. Friedl and D. Varro', "Graph Transformation in Relational Databases " vol. 127, pp. 167-180 30, March, 2005.

[8] G. Taentzer, K. Ehrig, E. Guerra, J.d. Lara, L. Lengyel, T. Levendovszky, U. Prange, D. Varro' and S.Varro Gyapay, "Model Transformation by Graph Transformation: A Comparative Study," in *Satellite Event of MoDELS 2005* Montego Bay, Jamaica, 2005.

[9] B. Appukuttan, T. Clark, S. Reddy, L. Tratt and R. Venkatesh, "A model driven approach to building implementable model transformations," in *Workshop in Software Model Engineering* San Francisco, USA, October, 2003.

[10] J. M. Vara, B. Vela, J. M.C. Barca and E. Marcos, "Model Transformation for Object-Relational Database Development," ACM, pp. 1012-1019, 2007.

***Authors**: Farzaneh Mahdian,Isalmic Azad University, Tuyserkan branch, Email: far_mahdian@yahoo.com. Dr.Vahid Rafe, Arak University, Email:rafe@iust.ac.ir. Dr. Shahram Jamali, university of Moahghegh Ardabili, Ardabil, Iran, E-mail: jamali@iust.ac.ir.*