A BRIEF ANALYSIS OF SOFTWARE TOOLS FOR LOGICAL DATABASE DESIGN

V. Cotelea, PhD, assoc. professor
Academy of Economic Studies of Moldova

Integrated database design for a complex field of interest and usually large data volume is a difficult task. The existence of a design methodology allowed the emergence of assisted systems to develop software integrated with data dictionaries. Such tool CASE (*Computer Aided System Engineering*) has emerged - systems for structured database design and related information systems oriented on data models created in different DBMS.

Thus, two main directions prevail in development of CASE in design technologies: CASE systems for design of the database itself (so-called Upper-CASE) and integrated tools for assisting both database design and applications design that use them.

Often, integration of functions leads to a strong accretion of CASE system with a DBMS. For example, the CASE system for mapping the conceptual model into logical model is often done for predetermined DBMS.

Another fact is related to integration problem, which can occur when portable database is designed on different computers platforms, on different operating systems, DBMS's and even data models.

Methodologies for developing design tools and information systems are usually classified according to areas or features. However, potential designers are more concerned about situations where different approaches are appropriate. In paper [2], five classes of situations are identified: (1) well-structured problems in cases with well-defined tasks and clear requirements, (2) structured problems in cases with clear objectives but uncertain user requirements, (3) unstructured problems in cases with unclear objectives, (4) cases where there is a high user interaction with the system and (5) complex problems in situations that combine two or more (1) - (4) classes. A multiview approach must be taken in developing information systems.

For example, paper [14] proposes a number of techniques to be used in teaching database design. The common notations for the entity-relationship diagram are discussed. It is developed for this purpose an entity-relationship diagram notation adapted from UML conceptual modeling language, which facilitates student learning of the database design process. The authors present a specific step by step process for representing entity-relationship components such as tables and for normalizing the resulting set of tables.

The design process should be interactive one. It is good to be a web-based tool. The tool described in [10], for example, is suitable for relational data modeling in analysis and design systems, and training of database designers.

Under the leadership of Gerritsen there have been implemented several tools to perform various tasks associated with database design. In [7] an example to illustrate the capabilities of these integrating tools is presented. There are also described future plans regarding integration of tools for providing database designer a more complete and consistent service.

Thus, Gerritsen has implemented DESIGNER tool that automates large scale logical design of the database and DBD-DSS (Decision Support Data Base Design System), which assists the physical design of the database and dynamic restructuring. In addition, an optimizing constraints model was developed, which captures many decisions related to physical design. All proposed instruments use the terminology, data structures, access methods and concepts already widespread in this area.

A program for the normalization of relations that is written in Prolog has several advantages relative to programs written in conventional programming languages, notably, conciseness and clarity. The program presented in [4] implements several normalization algorithms and is suitable for the interactive design of small database applications and as a teaching aid.

Bitton and other authors [3] propose an expert tool, named DBE, which produces knowledge in relational design theory and query optimization automatically and transparently available to the database designer. This tool is a system with an interactive, graphical interface that uses examples to guide the designer through several phases of logical and physical database design. Logical design is based on example relations, and physical on example queries. The example relations are automatically generated by the system. They

contain sample data and satisfy the data dependencies that the designer specifies with the assistance of the expert tool. The example queries and their expected frequency are specified by the designer, using graphically displayed skeleton queries. The system generates a physical design scheme that optimizes the mix of queries expected by the designer, and computes a performance forecast. Both example relations and example queries can be modified by the designer, until the DBE generate a satisfactory design.

Schlimmer [13] has developed a tool that provides a set of electronic forms for editing the views and naturally displays dependency information from the underlying database. This paper explores the possibility that there may be multiple keys, some unanticipated by the database designers. Like the pre-specified key, these additional keys may be useful for database indexing or for conversion to a normal form. These keys are superfluous, however are convenient if a user wishes to add data to a projection of the database.

By searching for a key that most closely matches the attributes requested in the projection, the tool allows the user to edit a view that includes the minimum number of additional attributes. In the slowly emerging field of electronic forms, users may wish to automatically generate a form corresponding to a projection of a database. Identification of all keys in a database becomes a simple task and the obtained information is used in building normal forms.

New application domains in data processing environments pose new requirements on the methodologies, techniques and tools used to design them. The applications' semantics should be fully represented at an increasingly high level, and the representation should be subject to rigorous validation and verification. In [15] is presented a semantic representation framework (including the language, methods and tools) for design of data-processing applications. The new features of the framework include a small number of precisely defined domain-independent concepts, high-level possibilities for describing behavioral semantics (methods and constraints) and the validation and verification tools included in the framework.

The paper [8] develops a method that maps an enhanced Entity-Relationship schema into a relational schema and normalizes the latter into inclusion normal form. Unlike classical normalization that characterizes individual relations only, inclusion normal form takes interrelational redundancies into account. In [8] is described a Prolog implementation of the method, developed in the context of a CASE shell for software development.

Akehurst and others argued in [1] that the normalization process can be automated by using a declarative approach to the specification of the normalization rules and a precisely defined transformation over a meta-model of a database system design language. A tool supporting the normalization of database system designs can subsequently be developed providing an invaluable aid to the software designer.

Douglas and Barker in [6] describe an intelligent tool for helping to teach the principles of database design. Presented software uses Prolog language to implement a training tool with which the concepts of dependency theory and the normalization process can be explored. The users are able to build their own learning environment and can develop their understanding of the material at a pace that is individually controlled.

In [9] an automated tool DBLint is described, it is used for analysis of database designs. DBLint provides a consistent and easy to maintain database project by identifying bad design models.

DBLint contains 46 predefined rules which analyze both metadata and data necessary for designing database objects. The rules are configurable and can be adjusted to the specific needs of the database administrator. In addition, the system provides a rules interface, so that designer can create his own rules, this way DBLint can be customized to the designer requirements.

At the output the system generates a report containing a detailed description of all found aspects and a description of the structure of the analyzed database. In this way the context with problems can be easily found and understood. The report also contains a score summarizing the overall quality of design, based on the shortcomings discovered rules.

DBLint is available in an online version, where each can send SQL scripts and get an answer instantly, or DBLint can also be downloaded onto one's own computer.

The paper [12] presents the main features and functionality of a new version IIS*Case R.6.21 (Integrated Information Systems* Case) developed in Java environment.

The IIS*Case is a CASE tool, based on the concept of "form type" and supports conceptual modeling of a database schema. Moreover, the system generates subschemes in the third normal form and performs an automatic integration into a relational schema. The system provides an automatic and intelligent support for complex and

highly formalized design and programming tasks. IIS*Case uses specialized algorithms for testing the consistency of constraints defined in the database schema and the subschemes. The system assists designers to review and validate the results obtained after each step of the design process.

A very important concept in the relational model is the concept of dependencies, especially functional dependencies. It is proven that functional dependencies can be represented by formulae of prepositional or predicate calculus. There are several systems of transforming functional dependencies into a logic system known in specialized literature, but all of them have one serious drawback: they do not have a form that is appropriate for reasoning about normalization. Lovrencic and others describe in [11] a new approach to the process of transforming functional dependencies into predicate calculus. The system presented in this paper is designed in the way to be appropriate for normalization, reasoning about it, as well as for the building a system for automated normalization of databases.

In [16] a complete interactive tool, named JmathNorm, is described, for relational database normalization using Mathematica. The developed tool can be used for real-time database design as well as an aid in teaching fundamental concepts of database normalization. JMathNorm also supports interactive use of modules for experimenting the fundamental set operations such as closure, and full closure together with modules to obtain the minimal cover of the functional dependency set and testing an attribute for a candidate key. JMathNorm's GUI interface is written in Java and utilizes Mathematica's JLink facility to drive the Mathematica kernel.

Dhabe and others introduce in paper [5] an Articulated Entity Relationship diagram, which is an extension of Entity Relationship diagram to accommodate the functional dependency information as its integral part for complete automation of normalization. The proposed diagrams are capable of accommodating complete information about the entities required for normalization up to Boyce/Codd normal form including their attributes, relationships and functional dependencies holds on them.

CONCLUSIONS

The tools for database design developed so far were focused mainly on the normalization. There is a complete lack of the systems that would perform analysis of the existing database design. This restrains the development of the existing information systems and their adaptation to new requirements of today. In addition, the tools are intended for a more didactic training. The algorithms used are the classical ones (of high complexity), which are applicable only on laboratory examples and may not serve real database design.

Therefore, more research on the development of automated design tools is needed, which would include solutions to problems related to automatic design. Furthermore, the investigations should be aimed at developing tools that would adopt the trends that are currently present in development systems: a very clear separation of logical and graphical components, as well as separation of the component implementing the algorithms and logical component. They can be developed separately, by separate teams, in different languages. Today, when developed systems are complex and the user must be removed from processes, the declarative languages can be applied to implement the algorithms. The tool must be adaptable, easy to attach new modules and create new contact interfaces with the designer.

Bibliography

1. Akehurst D.H., Bordbar B., Rodgers P.J., Dalgliesh N.T.G. *Automatic Normalization via Metamodelling, Proc. of the ASE 2002 Workshop on Declarative Meta Programming to Support Software Development, 2002, 167-172.*

2. Avison D. E., Fitzgerald G. *Information Systems Development: Methodologies, Techniques and Tools, 3rd Ed., Lodon, UK: McGraw Hill, 2002. 608 p.*

3. Bitton Dina, Millman Jeffrey C., Torgersen Solveig. *DBE: An expert tool for database design. Lecture Notes in Computer Science, V.498, 1991, p.240-263.*

4. Ceri S., Gottlob G. *Normalization of relations and PROLOG. Communications of the ACM, V.29, N.6, 1986, p. 524-544.*

5. Dhabe P. S., Patwardhan M. S., Deshpande Asavari A., Dhore M. L., Barbadekar B.V., Abhyankar H. K.. *Articulated Entity Relationship (AER) Diagram for Complete Automation of Relational Database Normalization. International Journal of Database Management Systems (IJDMS) , V.2, N.2, 2010, p.84-100.*

6. Douglas Paul, Barker Steve. *A Logic Programming E-Learning Tool For Teaching*

Database Dependency Theory. Proc. of the First International Workshop on Teaching Logic Programming: TeachLP 2004, Saint Malo, September 8-9, 2004, p.71-80.

7. Gerritsen Rob. *Tools for the automation of database design. Lecture Notes in Computer Science, V.132, 1982, p.72-86.*

8. Kolp Manuel, Zimanyi Esteban. *Enhanced ER to relational mapping and interrelational normalization. Information and Software Technology, V.42, N.15, 2000, p.1057-1073.*

9. ***Krogh Benjamin, Weisberg Andreas, Bested Morten.*** *DBLint: A Tool for Automated Analysis of Database Design. 2011, 33 p., http://projekter.aau.dk/projekter/files/43732470/final.pdf (vizited 12.11.2011).*

10. Kung H.J., Tung H.L. *A Web-based tool to enhance teaching/learning database normalization. Proceedings of the Southern Association for Information Systems Conference, 2006, p. 251-258.*

11. Lovrencic A., Cubrilo M., Kisasondi T. *Modeling Functional Dependencies in Databases using Mathematical Logic. Proc. of the 11th International Conference on Intelligent Engineering Systems. INES 2007, June 29-July 2, 2007, p.307-312.*

12. Pavićević Jelena, Luković Ivan, Mogin Pavle, Ristić Sonja, *Resolving Database Constraint Collisions Using IIS*CASE Tool. Journal of information and organizational sciences, V.31, N.1, 2007, p.187-206.*

13. Schlimmer Jeffrey C. *Using Learned Dependencies to Automatically Construct Sufficient and Sensible Editing Views. Proc. of the Knowledge Discovery in Databases Workshop, 1993, p.186-196.*

14. Thompson C. B., Sward K. *Modeling and teaching technique for conceptual and logical relational database design. Journal of Medical Systems, V.29, N.5, 2005, p. 513-525.*

15. van Keulen M., Skowronek J.,. Apers P. M. G, Balsters H., Blanken H. M., de By R. A., Flokstra J. *A Framework for Representation, Validation and Implementation of Database Application Semantics. Proceeding DS-6 Proc. of the Sixth IFIP TC-2 Working Conference on Data Semantics: Database Applications Semantics, Stone Mountain, Atlanta, Georgia, USA, May 30 - June 2, 1995, p.526-546.*

16. Yazici Ali, Karakaya Ziya. JMathNorm*: A Database Normalization Tool Using Mathematica. Lecture Notes in Computer Science, V.4488, 2007, p.186-193.*

Recommended for publication: 17.08.2011