Knowledge-Based Systems as Database Design Tools: A Comparative Study

W. Amber Lo
Millersville University &
Knowledge-Based Systems, Inc., USA

Joobin Choobineh
Texas A&M University, USA

A database is a collection of related data that represents some relevant reality. The task of designing a database has traditionally been performed manually by database designers. The design process turns informal end-user requirements into the design of static database structures, specification of integrity rules, and (may include) specification of dynamic aspects of data (transactions and queries to be made). The four major steps of database design are Requirements Collection, Conceptual Design, Logical Design, and Physical Design. This process is complex and error-prone. With the maturity of AI techniques, there is a significant potential to automate parts or all of the design process by developing knowledge-based systems as design tools.

The following terms will be used throughout the paper. A "database design methodology" is a system of principles, procedures, techniques, rules, data models, tools, documentation, planning, management, control, and evaluation applied to the entire design process. A methodology should describe each of the above components in detail (Maddison et al. 1983, p. 4). A "database design technique" is a systematic procedure by which a complex task within a step of the database design process is performed. A "data model" is a set of logical concepts that can be used to describe the structure of a database. It should consist of two parts, a notation for describing data and a set of operations used to manipulate that data (Ullman, 1988, p. 32). A data model usually is not as comprehensive as a methodology. It has a set of notations and a method of using it. However, it usually lacks a unified step-by-step guideline on how to use the concepts to represent a database structure. A “database design tool” is computer software used to perform or assist in one or more of the step(s) of the database design process. A tool is based on a data model or design technique. This can enhance the validity and uniformity of the design.

The objectives of this paper are threefold. First, to define desirable features of a knowledge-based database design tool. Second, to compare 23 existing knowledge-based systems so as to provide an overview and evaluation of the current state of research. Third, to identify future research directions in the development of computer-aided software engineering (CASE) tools for database design. Desirable database design tool features will be reviewed in the next section. Based on this survey, we
will conclude the paper by presenting a discussion on research progress and future research directions in section four.

**Desirable Features of Knowledge-Based Database Design Tools**

Based on published literature, we have chosen four distinct sets of desirable features of knowledge-based database design tools. These are database design support, tool flexibility, knowledge-based system features, and implementation features. A summary of these are presented in Figure 1.

The extent of usefulness of database design support includes the number of design steps covered, support for view integration, completeness of the output produced within the steps covered, the data models used, and the extent of the tool’s support of database design. The more steps a tool can automate, the more comprehensive it is. View integration support is important because, in a multi-user database environment, views of different user groups must first be documented and then integrated. Completeness of output refers to the fact that the more complete the output is, the more useful the system would be to the tool users. Data models supported by a design tool are the theoretical bases of that tool. In order to be rigorous and produce acceptable and reproducible output, a tool must be well grounded in some database design theory. The extent of a tool’s support of the design process depends on the levels of design expertise possessed by the tool, which in turn may determine its users. Targeted users, with a given level of database design expertise, are an important factor in determining the design of the tool, the user interface, the amount of expertise to reside in the knowledge base, and the role of the system in supporting database design activities.

Tool flexibility issues include application domain independence, data model independence, and maintenance features. A flexible tool should be able to aid in designing databases for any application domain. A tool is application domain independent if it can be used for designing databases for any domain. As design techniques and data models may change over time, an ideal tool should be flexible enough to facilitate the incorporation of these future changes. A tool is independent of any data model if its knowledge base is built in a manner that it can accept and use a new data model without extensive modifications such that existing data models can be updated and new ones can be added. With more than one kind of data model residing in the knowledge base, a tool user can make use of different ones for the same application and compare the output produced. The comparison enables the designer to obtain deeper insights into the nature of the application and/or it will help in deciding about a proper data model if that decision has not yet been made. A data model independent tool is more difficult to develop. More research and development is required because meta-level knowledge to accept new models is needed. Meta-level knowledge here is the knowledge about the proper way of defining a data model so that the constructs and rules of using them are rigorous and technically sound. With such tools, database design experts can enter the specification of any data model into the system and database designers can make use of them to design databases. The last flexibility issue is tool maintenance. Maintenance facilities are special components of a knowledge-based system that are written for future reprogramming of the system. An example is a facility to enable insertion of a rule into a rule base through a menu-driven sub-system instead of direct insertion into the source code. A tool that includes maintenance features is more flexible and can save substantial amount of human time and effort in the long run.

Knowledge-based system features include knowledge representation, inference mechanism, and explanation. Knowledge representation refers to the format in which a system stores its knowledge. Two important inference issues are the problem solving approach used and whether a system can incorporate uncertainties. From a tool user’s point of view, a very important knowledge-based system feature that a design tool should have is the ability to explain its actions and answer questions about database design. The availability of this feature is an important advantage of using knowledge-based systems to overcome the black-box problem.

Three issues regarding implementation that are discussed here are development environment, user interface, and system testing. Development environments could vary from programming languages that can be run on general-purpose machines to knowledge-based systems that only run on specialized machines. User interface should be flexible and friendly. A tool is flexible if, in order to communicate the same idea, it has different user interfaces for different kinds of tool users. A tool user that is very familiar with a tool may find command-based interface more direct and convenient than having to go through several levels of menus to perform a task. On the other hand, menu-driven or window-based interfaces can guide a novice tool user while he/she is learning to use the
Related Content

A Combined GA-Fuzzy Classification System for Mining Gene Expression Databases
[www.igi-global.com/chapter/combined-fuzzy-classification-system-mining/44384?camid=4v1a](www.igi-global.com/chapter/combined-fuzzy-classification-system-mining/44384?camid=4v1a)

Kernelized Database Systems Security
[www.igi-global.com/chapter/kernelized-database-systems-security/11164?camid=4v1a](www.igi-global.com/chapter/kernelized-database-systems-security/11164?camid=4v1a)

MILPRIT: A Constraint-Based Algorithm for Mining Temporal Relational Patterns
[www.igi-global.com/chapter/milprit-constraint-based-algorithm-mining/7966?camid=4v1a](www.igi-global.com/chapter/milprit-constraint-based-algorithm-mining/7966?camid=4v1a)

Roles of Resource and Data Contention on the Performance of Replicated Distributed Database Systems
[www.igi-global.com/article/roles-resource-data-contention-performance/51115?camid=4v1a](www.igi-global.com/article/roles-resource-data-contention-performance/51115?camid=4v1a)