Real World Knowledge for Databases

Veda C. Storey
University of Rochester

A number of expert systems have been developed recently that attempt to automate the database design process. One of the well-known problems with these and other types of expert systems is that, although they possess a high degree of expertise in their specific application areas, they usually know very little about anything outside those areas or about how the real world functions. This results in the need to ask the user many questions that a human expert would not need to ask about an application. It has been suggested that it would be beneficial for a database design system to be able to acquire general, or real world, knowledge over time as it is applied from one design session to another. Before this can happen, however, an understanding is required of what constitutes real world knowledge as well as how it can benefit database design systems. In this paper, various types of knowledge are examined and organized into a framework. The framework is then used to show how real world knowledge can be incorporated into database design expert systems.

One of the key factors that distinguish computer-based expert systems from human experts is that the latter apply to a problem, not only their specialized expertise, but also their general knowledge of the world and experience gained in previous projects. By comparison, expert systems are often described as having “narrow but deep” knowledge. They know a great deal about the specific areas in which they are expert, but little or nothing about anything outside these areas. This general knowledge about the way the world works is often referred to as commonsense, or real world, knowledge, and the lack of it is particularly noticeable for the database design task.

Databases are usually designed by “experts” who have detailed knowledge of a design methodology as well as the benefit of experience gained from previous designs. However, because these experts are called upon to work in many diverse areas, they may not know very many details about the specific application for which a database is required. In order to overcome this difficulty, the expert interacts with a knowledgeable user who supplies the application expertise. The success of this process depends both upon how well the user can articulate his or her requirements and how well the designer is able to extract and represent these requirements in a database. A designer usually knows something about a specific application area or is able to apply his or her general knowledge about an application to the design task.

Recently, a number of knowledge-based, or expert, systems have been developed that attempt to automate the database design process. Although they contain considerable knowledge about database design in their knowledge bases, these systems usually know very little about how the real world functions, and are not able to augment their knowledge based upon experience, as a human designer would do. This often results in the need to ask a user many questions that a human expert would not need to ask about the application [Storey and Goldstein, 1991]. It would, thus, be beneficial for a database design system to be able to acquire and use real world knowledge as it is applied from one design session to another. Before this can happen, an understanding is first required of what constitutes real world knowledge and how it can benefit and be incorporated into database design systems.

This research is concerned with database design systems for business applications. Specifically, the objectives of this
paper are to:

• define “real world knowledge” within the database context;

• discuss the need for acquiring real world knowledge in database design systems;

• identify various types of knowledge that are important for database design systems and organize them into a Knowledge Framework in order to illustrate how they might be used.

The paper is divided into five sections. The next section discusses knowledge-based systems for database design. The following section illustrates the need for capturing real world knowledge, and discusses previous research. Then, various categories of real world knowledge are examined and organized into a framework. A prototype system, based on this framework, is briefly presented. Finally, a summary and concluding remarks are given.

Knowledge-Based Systems

In the traditional approach to database design, the design task usually has been carried out by a design expert who obtains information about a user’s data needs through interviewing, examining existing documents and systems, and other means. The main difficulties with this approach are that it relies on expert database designers who are scarce, and the design of a database is done by someone who is unfamiliar with the application domain, instead of by end-users who best understand the application. In an attempt to overcome some of these problems, a number of automated systems have been developed to assist the database design process (for overviews of these, see Lloyd-Williams [1991]; Ram [1989]; Storey and Goldstein [1991]). The objectives of these systems are either to replace a database designer or to provide “intelligent” assistance to the task. Some of these systems can be classified as knowledge-based or expert systems; others are simply clerical tools. For the purposes of this paper, “knowledge-based system” is employed as a generic term to describe systems that use techniques from Artificial Intelligence (for example, knowledge representation or reasoning techniques). To be classified as “expert”, a system should work in some sense as would a human expert or accomplish its task as well as a human expert, regardless of how it does so [Szolovits, 1986].

The desired advantages of a knowledge-based or expert systems approach to database design are that such systems can: help to compensate for the scarcity of human experts; serve as “intelligent” consultants or assistants to a database designer; and, for those systems that interact successfully with the end-user of a database, allow the end-user, who should understand the application best, to become the designer of his or her own database [Storey and Goldstein, 1991].

Current Systems

Examples of knowledge-based systems that attempt either to aid or replace a human database designer include the following.

• The Intelligent Interview System (IIS) [Kawaguchi et al., 1986] tries to obtain a database design while learning both about the design application and about database design, in general, through interviewing the end-user of a database. It tries to prompt the user for missing requirements.

• The View Creation System [Storey, 1988; Storey and Goldstein, 1988; 1990a; 1990b] is an interactive expert system that engages the end-user of a database in a dialogue designed to elicit his or her information requirements and represent them in a relational model.

• The Expert Database Design System (EDDS) [Choobineh, 1985; Choobineh et al., 1988] is an expert system for conceptual database design that creates an Entity-Relationship diagram by analyzing a collection of forms.

• The Automated View Integration System (AVIS) [Wagner, 1989] integrates related user views into a single conceptual schema, based on heuristics for resolving name, construct, context, and meaning conflicts.

• Modeller [Tauzovich, 1989; 1990] is an expert system for conceptual design. It uses a rich taxonomy of modelling concepts and a variety of rules to reason about logical consequences of assertions supplied by the user as they are added to an evolving Entity-Relationship model.

• The Système Expert en Conception de Systèmes d’Information (SECSI) [Bouzeghoub et al., 1985; Bouzeghoub and Metais, 1986] is an intelligent tool to assist end-users or database designers in designing a database. It produces a set of normalized relations (Fourth Normal Form) along with a set of integrity constraints.

• EXIS [Yasdi and Ziarko, 1987] assists a user in designing a conceptual database schema. It uses a learning algorithm to induce functional dependencies. EXIS tries to “learn” about database design, in general, by storing sample designs in an “example base”, along with the rules that were used to generate them. Some implementation has been carried out, with more work planned to do so. The objective is for EXIS to learn enough about database design so that it can eventually replace a designer.
Related Content

Cost Modeling and Range Estimation for Top-k Retrieval in Relational Databases
[www.igi-global.com/chapter/cost-modeling-range-estimation-top/52960?camid=4v1a](www.igi-global.com/chapter/cost-modeling-range-estimation-top/52960?camid=4v1a)

Accommodating Hierarchies in Relational Databases
[www.igi-global.com/chapter/accommodating-hierarchies-relational-databases/8276?camid=4v1a](www.igi-global.com/chapter/accommodating-hierarchies-relational-databases/8276?camid=4v1a)

A Study of Open Source Software Development from Control Perspective
[www.igi-global.com/article/study-open-source-software-development/49722?camid=4v1a](www.igi-global.com/article/study-open-source-software-development/49722?camid=4v1a)

A Conceptual Design Methodology for Fuzzy Relational Databases
[www.igi-global.com/article/conceptual-design-methodology-fuzzy-relational/3332?camid=4v1a](www.igi-global.com/article/conceptual-design-methodology-fuzzy-relational/3332?camid=4v1a)