Knowledge Bases and Database Engineering

D. Stott Parker (chairperson)
Dept. of Computer Science, UCLA,
Los Angeles CA 90024, USA

Stefano Ceri
Politecnico di Milano, Piazza Leonardo Da Vinci 32,
20133 Milano, ITALIA

Robert Demolombe
ONERA-CERT-DERI, 2, Ave. Edouard Belin,
B.P. 4025, 31055 Toulouse CEDEX, FRANCE

Koichi Furukawa
Institute for New Generation Computer Technology,
4-28, Mita 1-Chome, Minato-ku, Tokyo 108, JAPAN

Masanobu Matsuo
Sumitomo Electric Industries, Ltd.
1-3, Shimaya 1-Chome, Konohana-ku, Osaka 554 JAPAN

Gio Wiederhold
Dept. of Computer Science, Stanford University,
Stanford CA 94305, USA

Economic pressures now encourage the engineering of comprehensive information management systems to control large volumes of heterogeneous information or 'knowledge'. These systems interface conventional database management systems with spreadsheets, documents, and rule bases. Storing large bases of knowledge, modeling the knowledge accurately, and accessing the knowledge conveniently become increasingly important when the information managed by an enterprise is a model of that enterprise.

The fields of Data Engineering, Knowledge Engineering, and Software Engineering overlap in the development of systems combining large volumes of data and knowledge. This panel is concerned with key issues in developing such systems. Some major problems in this area are listed here.

Although fairly advanced knowledge engineering tools are now available, knowledge-based systems are not well understood, and there is very little experience with large knowledge bases. *What functions must knowledge-based systems provide?* Current research papers suggest they should support access to type hierarchies, integrity management, and set-oriented recursive query processing, for example.

Knowledge and Data Engineering are fields of complex *performance tradeoffs*. Increases in flexibility or accuracy of knowledge representation can dramatically increase the complexity of knowledge access. Adding one function to a system can make other functions unacceptably slow, introduce redundancy in storage, etc.

In a recent workshop on knowledge engineering, participants from both industry and academe identified the lack of clear *design methodologies* as a major obstacle to successful use of the tools and development of systems, not to mention the education of designers and end-users on what KBMS can offer. *What methodologies can be used for designing knowledge-based systems?* Good design methodologies come from experience, and experience comes from bad designs.

Information systems of the future will have to communicate with information systems of the past, and different systems rest on different conceptual models (knowledge representation schemes). Integration requires some form of integration of their models. *Which kind of model — object-oriented, functional, or logic-based — is (dis)advantageous under which circumstances?* There is little consensus on how best to integrate models.

All conceptual models have weaknesses. For example, expert system models do not seem to scale as well as database models do. *Why is it so difficult to expand expert systems beyond a few hundred rules?* Even the best-known systems today are essentially prototypes with only a limited number of rules or facts. Also, *is there any difference at all between an expert system and a decision tree?*

Conceptual modeling schemes are evolving rapidly today to capture more information. These schemes have expanded from 'shallow' models of objects to 'deep' models of the behaviors of and constraints on those objects as well. *Are current DBMS models sufficiently rich to be the basis for knowledge-based systems, or are more elaborate models needed?* DBMS conceptual models seem shallow and not very extensible.

Logic Programming and Prolog extend relational databases with deduction and the ability to store complex objects such as schema, metadata, and constraints with database facts. It also provides an elegant and uniform way of implementing views, query languages, and null values. *When is Logic Programming the best strategy for extending DBMS to KBMS?* Systems like Prolog lack responsive query interfaces, and lack support for data processing concepts such as transactions, indexing, and integrity.

What are general methodologies for connecting KBMS and DBMS? General methods for binding these systems seem necessary if the area is to avoid remaining underdeveloped. Idiosyncracies of individual DBMS make such general integration challenging, but DBMS provide the technology for dealing with large collections of data, and this should not be re-invented.

Proceedings of the Twelfth International Conference on Very Large Data Bases **Kyoto, August, 1986**