in: Proc. 4th Brazilian Symposium on Data Bases, Campinas, April 1989

A Framework for an Efficient DBS-Support of Knowledge-Based Systems

F. - J. Leick, N. M. Mattos

University of Kaiserslautern, Department of Computer Science
P.O. Box 3049, 6750 Kaiserslautern, West Germany
e-mail : mattos@uklirb.uucp

1. Motivation

During the last few years a variety of Knowledge Based Systems (KS) has been developed as an application of techniques from the area of Artificial Intelligence (AI). When modeling real world applications, these systems are faced with problems of managing large amounts of knowledge, since virtual memory sizes are not large enough to store the corresponding Knowledge Bases (KB).

Realizing that existing Database Systems (DBS) provide features to fulfil this kind of requirement, research efforts have been done with the purpose of coupling KS and DBMS. So, the knowledge engineering tool KEE [FK85], for example, has been extended with a new component (the so-called KEEconnection [In87]), which enables it to store facts in an external DB to be managed by an independent relational DBS.

The coupling approach may perhaps solve KS limitations concerning virtual memory size, but it fails, however, to support KB management for several other reasons [Ma88b]. For example, when KB are maintained on secondary storage devices, operations on knowledge (e.g. inference) are so computationally intolerable, that coupling DBMS and KS yields very low performance [HMP87].

The solution to the DB deficiencies to support KS is to develop a new generation of systems, the so-called **Knowledge Base Management Systems (KBMS)**, aimed at the modeling and manipulation of knowledge as well as at its maintenance in very large, and possibly distributed, KB. Following this shift of viewpoint, we implemented at the University of Kaiserslautern a multi-layered prototypical KBMS, called KRISYS, which supports the above mentioned demands in an effective and efficient manner [Ma88b]. In this paper, we concentrate on the KRISYS component responsible for the KB management, thereby describing the framework provided by the system for an efficient processing of knowledge.

2. The Framework

Most of the performance difficulties of large KS result from the use of secondary storage. Further deficiencies may arise from the communication and transfer overhead between the used machines. When the KB is distributed or maintained in a remote server while the KS runs at a local workstation, knowledge has to be continuously extracted from the server and transferred to the workstation, where it is exploited for solving problems. The use of secondary storage as well as the server/workstation environment generate a so long path length when accessing KB contents, that these accesses turn out to be very time consuming [Ma86].

Exploitation of the application's locality

It is, therefore, desirable to have a mechanism that enables the reduction of the path length of the KS accesses, in order to improve performance. This is achieved by the KRISYS component, called **working-memory**, which temporarily stores and maintains objects of the KB in main memory, allowing the KS to reference these objects almost directly. The working-memory is a kind of application buffer, which offers very fast access to the stored objects. Consequently, KRISYS supports a processing model aimed at high locality of object references, thereby drastically reducing the path length when accessing KB objects.

Knowledge Independence

During the problem solving process, objects are transferred from the KB and placed into the working-memory. Certainly, it is not a task of the KS to worry about such knowledge transfer. KBMS should support the concept of **knowledge independence**, so that KS are unaware of the internal representation and storage of knowledge. Note that this (knowledge representation and storage) concealing principle is not provided by existing KS tools. In a KEE environment, for example, the KS must not only know whether its needed objects are stored in the external DB or not, but also how the mapping of knowledge to DB relations is carried out in order to process a KB. Moreover, this mapping can be quite difficult because the expressiveness of the knowledge representation model is much higher than the one of the DB model. Since knowledge structures are often mapped to several DB objects, it is not always possible to guarantee that changes on the knowledge structures will be appropriately reflected in the external DB. By treating knowledge as a kind of "window" to the DB contents, KEE presents the well known problems and restrictions of updates through views [Ma83, Ke81]. Finally, when constructing a KB, the knowledge engineer has to deal with two different environments, defining both knowledge structures and DB-schemas, as well as the mapping from one to the other.

For this reason, most present-day tools have to be classified as knowledge dependent. This means that the way in which the KB is organized in a storage device (i.e., DB and DB-schema) and the way in which it is accessed (i.e., the mapping) are both dictated by the requirements of the KS, and moreover, that the information about the organization and the access technique is built into the KS logic (i.e., embedded in their program's code). In such tools, it is impossible to change the storage structures (i.e., changes on the DB-schema) or access strategies (i.e., changes on the mapping) without affecting the KS more than likely drastically. Such a kind of knowledge dependence is particularly critical since computer systems should be continuously ameliorated, however, without affecting their application programs. Following this issue, KRISYS supports **knowledge independence**. It guarantees local confinement of all modifications and improvements throughout its lifetime, thereby isolating its applications from the internal representation and the storage of knowledge. Therefore, rather than having to explicitly inform our KBMS when knowledge is needed, KRISYS analyses the references of the KS to KB objects and automatically extracts these objects from the server, when they are not at hand in the workstation.

Exploitation of processing contexts

In general, objects stored in the working-memory are replaced by means of a LRU-strategy. An analysis of this strategy revealed however, that often KRISYS still has to perform many calls to its server component due to the very small granule of the unit of replacement (i.e. one object). The count of KB accesses increases especially after the KS starts a new phase of its problem solving process, when the majority of the needed objects is not found in the working-memory. Hence, it is very important to have ways to minimize such KB accesses. This can be achieved by supplying KRISYS with enough information about the KS access behavior, giving it more scope for optimization. The issue used in the system is to exploit the existence of the so-called **processing contexts** [HMP87, MW84], that are defined as the knowledge necessary to infer the specific goal of a phase of the problem solving process. Consequently, such processing contexts depend on the problem solving strategy being used by the KS. For example, KS exploiting the constraint-propagation strategy [St81] have contexts corresponding to the several constraints used to reduce the search space during the KS consultation. In the same manner, a problem-reduction strategy [Mc82] determines contexts for processing each partitioned subproblem, and generate-and-test [BF78] organizes KB contents according to the pruning activity (for an overview of existing strategies see [St83] or [St82]). For this reason, the knowledge engineer, as the one who knows the KS in fullest details, knows about the existence of such contexts. His task is to make known the existence of such contexts in order to supply KRISYS with valuable information fruitful for performance improvement. From the modeling point of view, defining contexts means making explicit the knowledge about the KS problem solving behavior, thereby improving KB semantics.

In principle, contexts are composed of several KB objects (in general of different types) and objects may be elements of several contexts. Since contexts are not fastened to constructs of the knowledge model provided by KRISYS [DM88, Ma88a], they may be specified or deleted at any time during KB construction or even dynamically during KS consultation. A dynamic definition of contexts is particularly important, because needed contexts are often established on the basis of the results of the preceding phases of the problem solving process.

Contexts are, therefore, the most important mechanism used by KRISYS to improve access efficiency. When defined during the KB construction, KRISYS can use special storage structures or clustering to optimize secondary storage accesses to these objects. But even when contexts are defined during KS consultation, our KBMS can use them to reduce KB accesses significantly. KRISYS exploits the existence of a context to generate a set-oriented access to the KB in order to fetch the objects of the context and store them into the working-memory, as soon as it is informed that a new processing phase of the KS will begin. So, most or perhaps all objects referenced during this phase are found in the working-memory and only few or no references to the KB are further necessary. At the end of the processing phase, its corresponding context is then discarded from the working-memory and the context requested by the following phase is loaded into it. By means of this set-oriented fetching and discarding of KB objects, the server component of KRISYS can better employ its optimization potential, thereby drastically reducing I/O and transfer overhead. Finally, the KS references to individual objects will be very efficiently supported, since the path length, when accessing the working-memory objects, is very short.

In closing, one may observe that there are two orthogonal ways of looking at a context. Viewed from the knowledge engineer, a context is a collection of objects needed by a specific phase of the problem solving process of the KS, i.e., the knowledge necessary to work out a particular problem. KRISYS, on the other hand, views it as a collection of objects which are brought into the working-memory by just one very efficient KB access. Following this approach, the knowledge engineer is still working in his framework, without being involved with internal KBMS aspects, although KRISYS is supplied with enough information for performance improvement [MM88]. In other words, the processing context approach fortifies the discussed support of knowledge independence.

3. Overall Architecture

Internally, the component of KRISYS responsable for the working-memory management is divided in three main subcomponents as shown in figure 1.

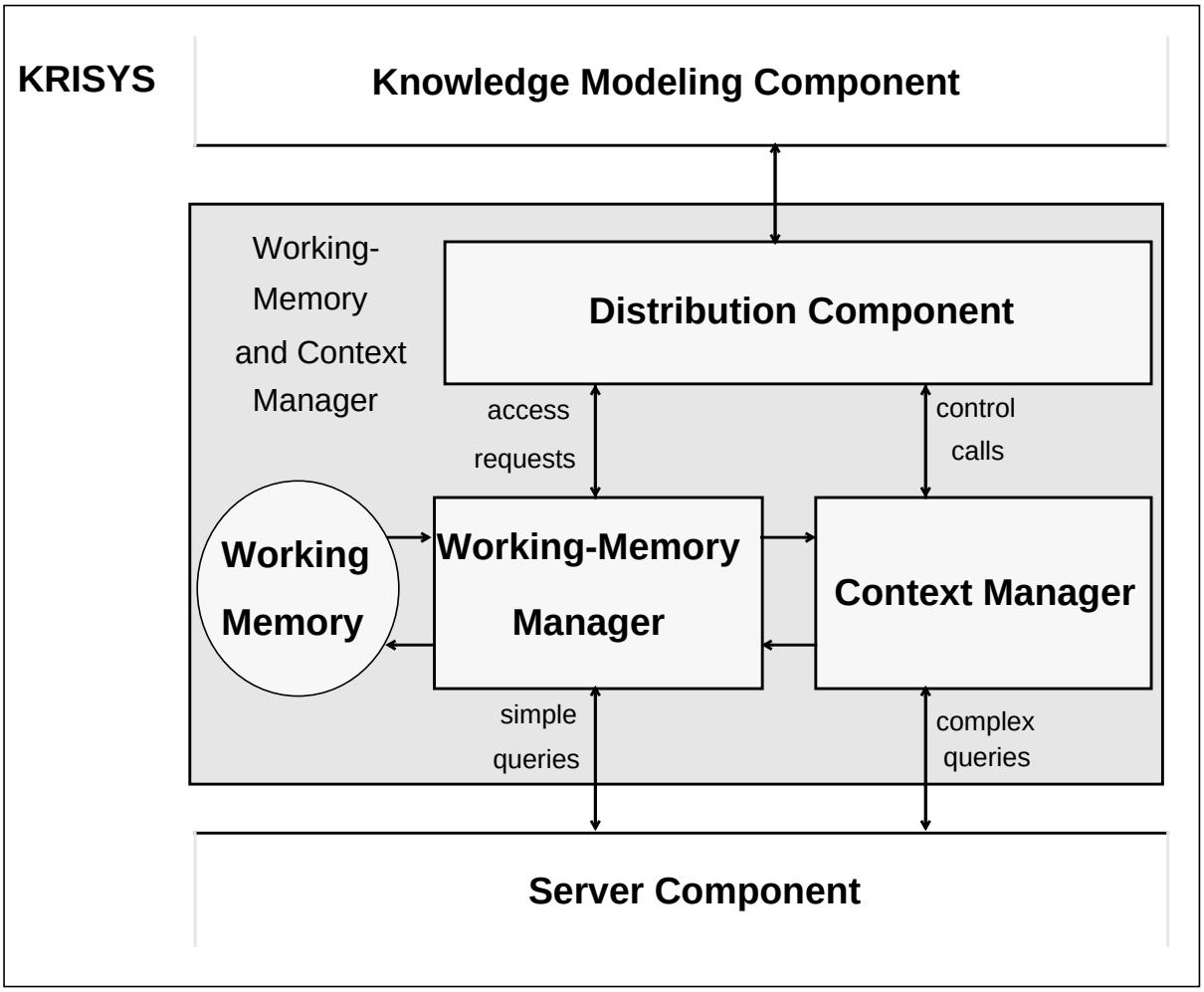

Figure 1: Structure of Working-Memory and Context Manager (WMCM)

The **distribution component** is responsible for the interpretation of the KS calls and the activation of the respective component. The **working-memory manager** carries out the access requests, sometimes generating simple queries (read/write accesses) to the server component of KRISYS when the requested ob-

ject is not found in the working-memory. The **context manager** is responsible for the execution of control calls, such as fetching or discarding of contexts. It carries them out by means of set-oriented DB-operations (complex queries) to extract the contents from the KB or to discard them from the working-memory. Therefore, the working-memory manager takes care of the replacement of individual objects, whereas the context manager takes care of the replacement of object sets (contexts).

In order to guarantee very fast access to the working-memory contents, the working-memory manager maintains two hash-tables, referring to the objects and to the attributes of the objects currently stored in the working-memory. Thus, it supports a direct access to attributes avoiding the usual sequential search within an object, which is in some cases (e.g., very complex objects) computationally intolerable. Both, objects and their attributes, are maintained in structures with variable length containers, supporting an efficient storage of KS objects, whose lengths are in general extremely different [Ma86].

4. Performance Comparison

The performance of our working-memory component has been compared with the performance of other coupling approaches between KS and DBS as shown in figure 2.

	Accesses to the KB (on secondary storage)	Duration of a Consultation in CPU-sec.	slow down factor with regard to main memory KS
main memory based KS	3574	164	1
KS - DBS direct coupling	3574	28105	~ 170
KS - DBS coupling with application buffer	572	2946	~ 18
Working-Memory Approach	93	643	~ 4

Figure 2 : Performance Comparison of Different Approaches for KB Management

In order to pursue this investigation, we have adapted some available KS to work on the different environments shown in the figure. As basis for this comparison, we use the main memory-based form of KS (i.e., when the whole KB is stored in main memory). Coupling DBS and KS directly seems to be the most inefficient alternative, due to the already discussed long path length of the application's accesses. This problem is eliminated by using an application buffer, which uses LRU as replacement strategy, to keep the most

recently used objects in the buffer. Since this approach extracts and discards only individual objects respectively from the KB and the buffer, very many calls to the DBS are still necessary, especially after changes of the processing phases. The last approach, which is based on the above described working-memory concept and implemented as part of the KBMS KRISYS, accomplishes an efficiency very close to the main memory one. (Further performance analyses can be found in [HMP87].)

5. Summary

In this paper, we described the framework provided by the KBMS KRISYS to support an efficient processing of large and even distributed KB.

KRISYS makes use of the application's locality to guarantee very fast accesses to the KB objects. It takes advantage of the knowledge about the access behavior of the KS, like for example the existence of processing contexts, to manage relevant objects in a main memory structure, called working-memory, which is maintained very close to the application, thereby drastically reducing I/O and transfer overhead. The definition of such processing contexts is executed by the knowledge engineer, which however keeps working in his original framework without being involved with internal aspects of KBMS. He only has to make known the existence of contexts as well as the phases of the KS problem solving process.

Further concepts (not discussed in this paper) of KRISYS pay attention to knowledge modeling and manipulation requirements [Ma88a, Ma88b]. The system has proven to be a very powerful tool for building KS, CAD applications, etc., thereby being considered a very fruitful research vehicle as well as a generic system for effective and efficient management of large KB.

Acknowledgments

We would like to thank the measurements performed by B. Rheinberger as well as his fruitful comments on the obtained results which contribute to improve the described issues. Also acknowledged are the careful reading and useful hints of S. Deßloch and B. Mitschang.

References

[BF78] Buchanan, B.G., Feigenbaum, E.A.: DENDRAL and Meta-DENDRAL: Their application dimension, in: Artificial Intelligence, Vol. 11, No. 1, 1978, pp. 5-24.

[DM88] Deßloch, S., Mattos, N. M.: KOALA and its Knowledge Model - An Application Interface for a New Generation of DBS, internal report, University of Kaiserslautern, 1988, submitted for publication.

[FK85] Fikes, R., Kehler, T.: The Role of Frame-based Representation in Reasoning, in: Communications of the ACM, Vol. 28, No. 9, Sept. 1985, pp. 904-920.

[HMP87] Härder, T., Mattos, N. M., Puppe, F. : On Coupling Database and Expert Systems (in German), in : State of the Art, Vol. 1, No.3, 1987, pp. 23 - 34.

[In87] IntelliCorp. Inc.: KEEconnection: A Bridge Between Databases and Knowledge Bases, IntelliCorp., Technical Article, 1987.

[Ke81] Keller, A. M. : Updates to Relational Databases Through Views Involving Joins, Research Report RJ 3282, IBM Research Laboratory, San Jose, California, 1981

[Ma83] Masunaga, Y. : A Relational Database View Update Translation Mechanism, Research Report RJ 3742, IBM Research Laboratory, San Jose, California, 1983

[Ma86] Mattos, N. M.: Concepts for Expert Systems and Database Systems Integration (in German), Research Report No. 162/86, University of Kaiserslautern, Computer Science Department, Kaiserslautern, 1986.

[Ma88a] Mattos, N. M.: Abstraction Concepts: the basis for data and knowledge modeling, in: 7th Int. Conf on Entity-Relationship Approach, Rom, Italy, Nov. 1988.

[Ma88b] Mattos, N. M.: KRISYS - A Multi-Layered Prototype KBMS Supporting Knowledge Independence, in : Proc. Int. Computer Science Conference - Artificial Intelligence : Theory and Application, Hong Kong, Dec 1988.

[MM88] Mattos, N. M., Michels, M.: Modeling Knowledge with KRISYS: the Design Process of Knowledge-Based Systems Reviewed, internal report, University of Kaiserslautern, 1988, submitted for publication

[Mc82] McDermott, J.: R1: A Rule-based Configurer of Computer Systems, in: Artificial Intelligence, Vol. 19, 1982, pp. 39-88.

[MW84] Missikoff, M., Wiederhold, G.: Towards a Unified Approach for Expert and Database Systems, in: Proc. of the First International Workshop on Expert Database Systems (ed. Kerschberg, L.), Kiawah Island, South California, October, 1984, pp. 186-206.

[St81] Stefik, M. J.: Planning with Constraints (MOLGEN: Part 1), in: Artificial Intelligence, Vol. 16, No. 2, 1981, pp. 111-140.

[St82] Stefik, M. J. et al.: The Organization of Expert Systems, A Tutorial, in: Artificial Intelligence, Vol. 18, 1982, pp. 135-173.

[St83] Stefik, M. J. et al.: The Architecture of Expert Systems, in: Building Expert Systems, Volume 1 (eds. Hayes-Roth, F., Waterman, D. A., Lenat, D. B.), Addison-Wesley Publishing Company, Reading, Mass., 1983, pp. 89-126.