DATABASE ACCESS REQUIREMENTS OF KNOWLEDGE-BASED SYSTEMS

Yannis Vassiliou
Jim Clifford
Matthias Jarke

April 1984

Center for Research on Information Systems
Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #74

GBA #84-49(CR)

Published in W. Kim, D. Reiner, D. Batory (eds.), Query Processing in Database Systems, Springer-Verlag, 1985.

DATABASE ACCESS REQUIREMENTS OF KNOWLEDGE-BASED SYSTEMS

Abstract

Knowledge bases constitute the core of those Artificial Intelligence programs which have come to be known as Expert Systems. An examination of the most dominant knowledge representation schemes used in these systems reveals that a knowledge base can, and possibly should, be described at several levels using different schemes, including those traditionally used in operational databases. This chapter provides evidence that solutions to the organization and access problem for very large knowledge bases require the employment of appropriate database management methods, at least for the lowest level of description -- the facts or data. We identify the database access requirements of knowledge-based or expert systems and then present four general architectural strategies for the design of expert systems that interact with databases, together with specific recommendations for their suitability in particular situations. An implementation of the most advanced and ambitious of these strategies is then discussed in some detail.

1.0 INTRODUCTION

Evidence for the successful application of Artificial Intelligence (AI) research is nowhere stronger than in the area of Knowledge-Based or Expert Systems (ES) [CLIF83]. In addition to being among the first AI systems which are finding their place in the commercial world, Expert Systems seem to lessen the controversies between differing AI research approaches, and even contribute to wider overlaps of AI with sciences like Philosophy or Cognitive Science.

Although a formal theory of Expert Systems is yet to be developed, there are some key common features that can be identified. As a computer system attempting to act like a human expert in a limited application domain, an ES shares similar goals with other, more traditional, computer application systems. What differentiates an ES from these traditional systems are its overall architecture, and, usually, its method of development. Expert Systems are typically developed in an incremental way. A small group of about three people, filling the three roles of application domain expert, programmer, and knowledge engineer, successively refines the ES to approximate the behavior of the expert.

This work was carried out as part of a joint study with the IBM Corporation.

It may be argued, however, that this approach is very similar to that taken in Information Systems Analysis and Design, especially when the technique of "prototyping" is used. The interested reader will find more information on this aspect of ES construction in [HAYE83] or other introductory articles on Expert Systems -- we shall not be concerned with the incremental nature of ES development in this chapter.

Overall architecture seems to be the most unusual aspect of Expert Systems. An often-quoted motto of ES researchers is that "in the knowledge lies the power." In consequence, an ES is based on two principles: the appropriate representation of the application domain knowledge, and the control of this knowledge. These two principles are embodied in the two top-level components of the ES architecture: a <u>knowledge base</u> and an <u>inference engine</u>. The application domain knowledge is represented in a <u>knowledge base</u>, which is further divided into two subcomponents: the <u>data level</u> (ground, specific facts), and the <u>knowledge level</u> (rules, general principles, or problem heuristics). This division of the ES knowledge base brings together two research threads in AI: declarative and procedural representation schemes.

The other architectural component of an ES, the control mechanism, is often termed an <u>inference engine</u>. The inference engine matches a problem description to the stored knowledge. This match can be done either to <u>analyze</u> a certain situation (e.g., in medical diagnosis) or to <u>synthesize</u> a solution for a specific problem (e.g., a computer configuration). Such an inference engine can be a pattern matcher, theorem prover, or network search mechanism customized for a particular expert system, or it may exist already in the compiler of a corresponding knowledge representation language such as OPS-5 [FORG80], Prolog [KOWA79], or EMYCIN [VANM79]). Even in the latter case, some additional control mechanism may be required to cut down the number of inferences to be made. Typical of the control techniques employed are state-space search, propagation of constraints, and problem reduction [GEVA82].

It is with respect to knowledge representation that strong parallels between AI and Database Management research can be drawn. Knowledge representation schemes in AI share the same objective with the data models which have been developed for Database Systems, namely, to represent data for an enterprise or "slice of reality." However, AI-based knowledge representations emphasize the richness, flexibility, and faithfulness of the representation, while the data models are limited from their realizations as Database Management Systems (DBMS), which emphasize efficient access and manipulation of a more permanently structured, stored representation of reality [1]. This difference is largely the result of the motivations in these two fields: modelling human reasoning processes on the one hand, and information management on the other.

After several years of parallel developments in AI and Database Management, it is now generally recognized that the two fields can benefit from exchanges of accumulated expertise in representation topics. This chapter considers the possible uses of Database Management principles and techniques for AI, and specifically for Expert Systems.

Following a brief introduction to AI knowledge-representation schemes and their use in current Expert Systems, the data-access requirements of an ES are identified. An analysis of four Expert System architectures for efficient access and storage of at least part of their knowledge base is then presented. These possible architectures are contrasted with a number of different data access requirements to present a set of specific recommendations for the

----

[1] [TSIC82] gives a detailed description of several data models, while [MYLO84] presents a comprehensive introduction to knowledge representation. Furthermore, both [WONG77] and [BROD84] examine the relationships between AI and DBMS representation schemes.

implementation of the data access component in an Expert System. Finally, some important research issues for the realization of these architectures are examined. Throughout the chapter we will illustrate various concepts and issues with examples taken from a life-insurance expert system currently under development at New York University.

2.0 KNOWLEDGE REPRESENTATION - AI AND DBMS

In this section, we examine AI knowledge representation strategies and relate them to their counterparts in Database Management Systems. A major conclusion that can be drawn is that it is possible to have multiple representations for the knowledge base of an ES. Furthermore, at least part of the knowledge base of an ES can be represented as a database under the control of a DBMS. Therefore, the designer of an ES has to consider the tradeoff between the choice of a specialized representation which allows for efficient access and storage of data, and the cost of providing translations between it and other representations more suitable for inference purposes.

According to the classification scheme in [NAU82], the ES's application domain knowledge can be described at two levels: data and knowledge. Generally, at the data level, the object types, their properties and relationships, together with the object instances are represented. This, in DBMS terms, corresponds to a database. At the knowledge level, rules and/or procedures and actions on the objects, together with meta-knowledge about the scope, limits, relevance, and importance of what is known are represented. There is no direct correspondence with DBMS concepts. Yet, some parallels can be drawn between representation at the knowledge level and data dictionary entries, coupled with externally written application programs for a database.

The separation of the knowledge base in data and knowledge levels should not be confused with the often-made distinction between declarative and procedural knowledge representation schemes [MYLO84]. In declarative representations, the knowledge base is a collection of declarative statements, while in the latter, a set of procedures constitute the knowledge base. But, a declarative representation scheme (e.g. first-order logic) can be used for the representation of the application domain both at the data and knowledge levels.

2.1 Knowledge: Domain Rules, Principles, And Heuristics

A knowledge base can be regarded as a collection of procedures expressed in some language (in AI systems, this has typically meant LISP). It is generally believed that the pattern-directed procedures of PLANNER [HEWI71] had a major influence in procedural knowledge representations [MYLO84]. Such procedures are not "called", as in ordinary programming languages, but are "activated" by the inference engine whenever the knowledge base is searched or modified.

Production Rules [WATE79] have become the most popular schemes among pattern-directed procedures. Each rule is a simple program with the format:

IF <condition> THEN <action>

where the condition is typically a conjunction of predicates, and the action activates other procedures which potentially change the state of the knowledge base. Production rules can be used to represent a variety of different types of inference, as illustrated in the following examples:

1. From SITUATION to APPROPRIATE ACTION:

```
IF evidence of diabetes is found
      THEN request a doctor's statement from applicant
```

2. From PREMISE to CONCLUSION:

    IF thiazide is present in the blood
        THEN applicant is taking high blood pressure medication

3. PROPERTY INHERITANCE

    IF applicant is female
        THEN applicant has a greater life-expectancy than a male

Production rules are typically used as description tools for problem-solving heuristics, replacing a more formal analysis of the problem into a deterministic algorithm. In this sense, the rules are thought of as "rules of thumb," incomplete but very useful guides to making decisions that cut down the size of the problem space being explored. These rules must be provided as input to an expert system by the human expert. This is usually done iteratively, perhaps by means of an interactive program that guides and prompts the expert to make this task easier, and which might also do some limited consistency checking. Rules have been proposed in some sense as a simulation of the cognitive behavior of human experts. Viewed in this light, rules can be seen not just as a neat formalism to represent expert knowledge in a computer but rather as a model of actual human behavior.

In combination with pattern-directed procedures, conditional probabilities and static descriptions of phenomena have been used in Expert Systems for the representation of knowledge (e.g., in Internist [POPL77]).

Given a suitable interpretation of logical formulas, first-order logic can also be used to represent procedural knowledge for the application domain. For example, assuming that P, Q, R are formulas, then the formula P AND Q -> R can be interpreted procedurally as: to show that R is true, first show that P and Q are true. For example, the logical formula:

```
applicant (X) AND age(X,Y) AND greater(Y,50) AND
visited-doctor(X,Date) AND difference-in-days(Date,Today,Z) AND
less(Z,90) --> needs-physical(X) OR needs-doctor-statement(X)
```

asserts that an applicant over 50 years old who has seen a doctor within the last three month must either submit a doctor's statement or submit to a physical examination.

A major problem with general first-order logic for knowledge representation is the difficulty in expressing control structures that efficiently guide the use of a large knowledge base. In the hope of reducing such problems, practical tools such as the logic programming language Prolog do not use full first-order logic, but only the subset known as definite (Horn) clauses. Furthermore, Horn clauses are interpreted in a procedural way reminiscent of the backward chaining of production rules, leading to a more efficient search process. However, the power of representation is reduced since only a subset of first-logic order is used. For instance, the example presented above is not a Horn clause.

2.2 Data: The Declarative Representation Of Facts

2.2.1 First-Order Logic. - Precise knowledge can be stated as assertions over objects that take the form of first-order predicates with functions and equality [KOWA79]. For example, unary predicates such as "male(X)" or "disease(X)" might be used to classify which of a set of objects were, respectively, males and diseases. Predicates of binary and higher degrees ("doctor-of(X,Y)", "premium(X,Y,Z)", etc.), and functions of arbitrary degrees (e.g. "age(X)") can also be defined. Simple facts are represented as instantiations of predicates, i.e., with actual values instead of variables (e.g. male(John), doctor-of(John, Elizabeth). Since logic is purely declarative it allows multiple uses of the same piece of knowledge, but, as mentioned above, it has been criticized for lack of organizational principles. There is a simple and direct correspondence between first-order logic and the relational model; [GALL78] provides a comprehensive view of this correspondence.

2.2.2 Networks. - Semantic networks [BRAC79] seem to be more popular in other AI applications (e.g., natural language processing) than in expert systems. Nevertheless, a number of expert systems rely on network formalisms, among them very large systems such as Internist [POPL77], Prospector [HART78], and Sophie [BROW81]. A portion of the knowledge about an insurance application and some of its related entities is represented in Figure 1. Networks are a natural and efficient way to organize knowledge: nodes describe objects, concepts, or situations whereas arcs define the relevant relationships. "Reasoning" in a network-based system corresponds to traversing the network along the arcs, or to pattern matching of problem descriptions with subnets; a large number of exact and heuristic mechanisms exist for these tasks. One advantage of semantic networks over logic (often referred to as "chunking") is that all knowledge about the object being described can be explicitly represented around the object, thus allowing for associative access to related knowledge. The disadvantages of the network approach stem from the lack of formal semantics for the structures, which makes verification of the correctness of reasoning very difficult. NETL [FAHL79] and KLONE [BRAC79] are examples of computer languages that have been developed for the access and manipulation of semantic networks. It is interesting to note that the networks of the CODASYL model of data can be viewed as a very simple form of semantic network.

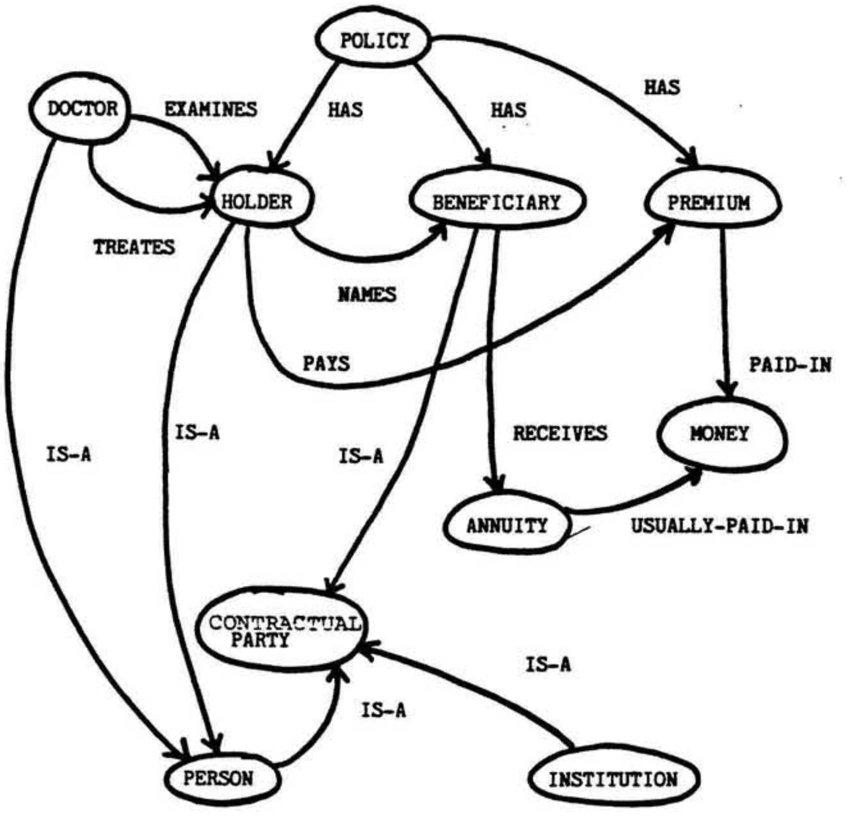
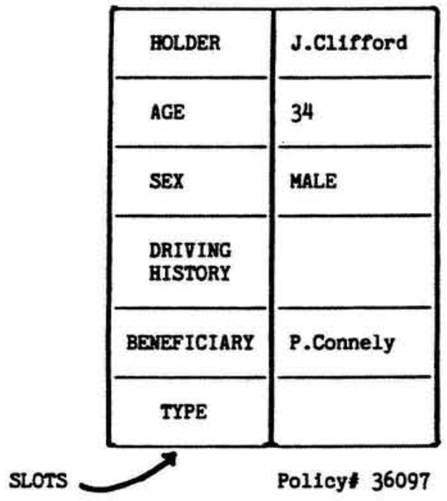


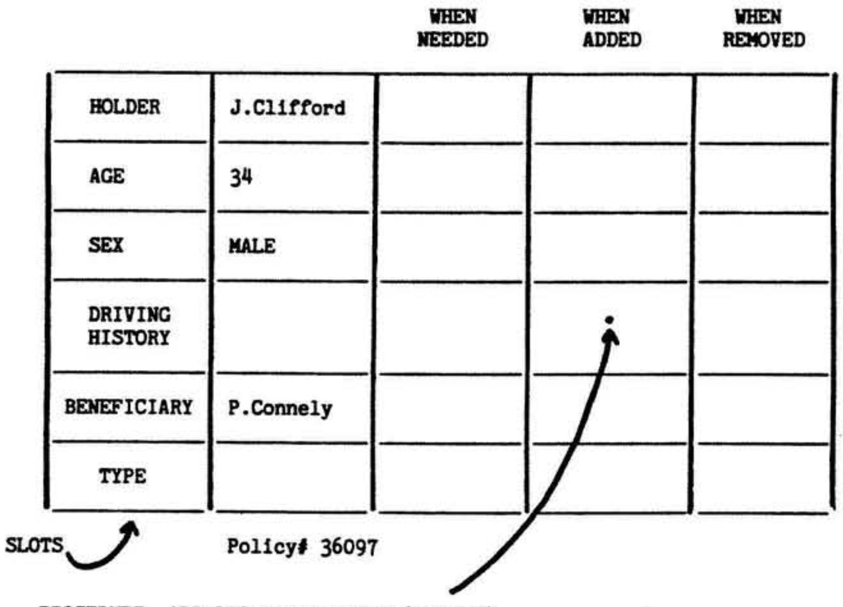
FIGURE 1: SEMANTIC NETWORK EXAMPLE

2.2.3 Frames. - Much knowledge is based upon experience, the expectations aroused from previous situations and the application of general concepts to a specific problem. Frames [MINS75, BOBR77] are a very general and powerful knowledge-structuring device that takes these psychological phenomena into account. They are templates for stereotypical situations, and provide a structure to such experiential knowledge by offering so-called slots which can be filled with type descriptions, default values, attached procedures, etc. The notion of attached procedures, in particular, allows for the development of general-purpose strategies for maintaining a frame-based knowledge representation. Figure 2a shows a simple instantiated frame representing information about a particular insurance policy. This particular frame might be expanded as in Figure 2b with slots to attach procedures that would



HOLDER	J.Clifford
AGE	34
SEX	MALE
DRIVING HISTORY	
BENEFICIARY	P.Connely
TYPE	

FIGURE 2a: FRAME EXAMPLE



		WHEN NEEDED	WHEN ADDED	WHEN REMOVED
HOLDER	J.Clifford			
AGE	34			
SEX	MALE			
DRIVING HISTORY			•	
BENEFICIARY	P.Connely			
TYPE				

```
PROCEDURE  ADD_DRIVING_HISTORY (POLICY)
           NOTIFY_UNDERWRITER (POLICY, POLICY.DRIVING_HISTORY
           EXPECT (POINT_ADJUSTMENT(POLICY) )
```

FIGURE 2b: FRAME EXAMPLE WITH PROCEDURAL ATTACHMENT

automatically be invoked when the knowledge base is modified by, e.g., addition or deletion of values in any of the slots of a frame. Like semantic networks, frames offer organizational principles for representing knowledge which are superior than the ones offered by logic formalisms. Several computer systems have been developed for the manipulation of these complex data structures. Among these, we mention KRL [BOBR77], STROBE [SMIT83], and FRL [GOLD77].

2.2.4 Multiple Knowledge Representations. - It has been argued [DAVI81] that no one of the knowledge representation methods is ideally suited for all tasks. In very complex systems requiring many sources of knowledge simultaneously (e.g., for speech recognition [ERMA80]), the elegance of uniformity may have to be sacrificed in favor of exploiting the benefits of multiple knowledge representations each tailored to a different subtask.

It is worth noting that it may often be difficult to distinguish between the data and knowledge levels in an Expert System. The same situation can be represented either at the data level, or at the knowledge level. Using the logic formalism of PROLOG, which combines procedural and declarative knowledge, we present an example of this phenomenon.

Consider the definitions of rating classifications given to prospective customers of an insurance company. The parameters for determining the rating are the customer's age, and the number of debit points accumulated from an analysis of other personal information. (For instance, a person gets "x" debit points if he or she smokes.) For example, a customer whose age is between 15 and 29, with an accumulated 20 points or fewer, is rated "preferred". With increasing age, the number of points that will disqualify a customer from getting a "preferred" rating decreases. Similar definitions exist for other classifications, such as "standard", "bad risk", "terrible risk", etc.

Procedurally, one can represent these classifications as production rules or logic clauses:

```
preferred  IF
     ((age between 15 and 29) AND (points between 0 and 200)) OR
     ((age between 30 and 39) AND (points between 0 and 150)) OR
     ((age between 40 and 49) AND (points between 0 and 100)) OR
     ((age between 50 and 99) AND (points between 0 and 50)).
```

with similar clauses for the other ratings.

Alternatively, these classifications can be given in a declarative way, by using logic assertions (unit clauses) with the general format: [2]

```
        rating(Min_Age,Max_Age,Min_Points,Max_Points,Rate_Class).
```

The "meaning" of such an assertion would reside in its interaction with the other predicates and rules in the knowledge base. These would have to be structured to give this assertion its interpretation as:

> A person between the ages of Min_Age and Max_Age, with a number of points between Min_Points and Max_Points, is classified in the specified Rate_Class.

For instance, the assertions:

----

[2] In this Prolog notation, predicates and constant values are represented in lower case, and variable names always start with an upper-case character.

```
        rating(15,29,0,200,'preferred').
        rating(30,39,0,150,'preferred').
        rating(40,49,0,100,'preferred').
        rating(50,99,0,50,'preferred').
```

and the clause:

```
        preferred  IF
          rating(Min_Age,Max_Age,Min_Points,Max_Points,'preferred').
```

define "preferred" customers. It can be seen that the above combines descriptions at the data and knowledge levels.

Notice the strong similarity between the above representation scheme with the relational model, where the last clause is similar to a relational "view" defined on the stored table RATING, as a selection where the Rate_Class attribute has value 'preferred'. This latter, declarative, representation is more flexible than the one that uses only procedural rules, in that it allows for multiple ways of viewing the ratings classifications. For instance, the question "what is the maximum number of points that a 50 year old customer may accumulate and still get a preferred rating" can be answered directly, given the declarative representation. Of course, depending on the application, this flexibility may not be necessary.

2.3 Remarks

It is fair to say that, in general, AI knowledge representation schemes are more powerful than their counterparts in database management, and in particular, the three popular data models: relational, hierarchical, and network. For instance, AI representation schemes embed inferencing capabilities. Using fixed inference rules, it is possible to deduce new facts from old ones.

However, the computer systems that have been developed for the manipulation of the basic objects (frames, semantic networks, etc.) in AI representation schemes lack most of the secondary storage management facilities, which are commonly offered in database management systems. For instance, features such as concurrency control, data security and protection, and, possibly most important, optimized access of data residing in secondary storage, are not part of AI systems. For example, Expert Systems typically load their knowledge base into main storage before the actual ES session begins. This may not have been an important limitation of previous or current Expert Systems; with very few exceptions, their application domains had no need of sophisticated DBMS mechanisms. It is only with the introduction of ESs into the commercial world that their data access requirements have begun to change. These requirements are identified and classified in the next section.

3.0 CLASSIFICATION OF ACCESS REQUIREMENTS

The data access requirements of Expert Systems can be classified along several dimensions. Among the most important are the volume of data needed to perform effectively, the origin of this data, and the timing of the decision as to which subset of the data is required. In this section we discuss these dimensions and illustrate them with examples.

Data Volume. - Historically, the knowledge bases in Expert Systems have been relatively small in size, small enough, in fact, to fit in main storage.

One of the largest knowledge bases for major Expert Systems is that of Internist. It is reported in [POPL83] to contain over 500 disease entities (each requiring several records), and 3,500 manifestations (history items, symptoms, physical signs, and laboratory data). [3] However, most ESs developed to date appear to have very modest data needs; for example, the initial production version of the expert system R1, which suggests a design for a computer system configuration, contained about 750 rules [MCDE81]. Even when new data is created through deductions or generate-and-test methods in Expert Systems, e.g., in [NICO83], and [STEF78], main storage is still sufficient to contain the knowledge base.

Recently though, it has become increasingly apparent that ESs may have to manage very large volumes of data. For instance, in CAD applications of Expert Systems large amounts of data are needed for the support of conclusions by an ES [LAFU83]. Also, for commercial applications of ES technology, very large operational databases will need to be consulted for more accurate ES operation [KUNI82, LAFU83, VASS83, JARK84a]. Of course, data volume is also relative to the size of the computer system and its main storage. In one of the first ESs implemented on a microcomputer, the LSI-11, the knowledge base contains data on scientific disciplines and development goals in Portugal, together with their interactions [PERE82]. The knowledge base in this system consists of a hierarchical set of tables representing correspondences between scientific disciplines and government development goals. Since these tables are quite extensive (roughly 26,500 "facts" are being represented), not all of the knowledge base in this system can reside in main storage. However, the storage management strategies developed in this case are appropriate only to the particular domain and are not readily generalized.

In cases like the above, main storage is insufficient--with virtual memories, you may never run out of space, but, eventually, will certainly run out of time.

Database Origin. - Most Expert Systems developed to date use their own database of facts, custom-made to best suit the application-specific requirements. Practically speaking, however, if ESs are to be applied in the commercial world they will need access to "external" data sources. These "external" data are typically operational databases managed by a DBMS and shared by other applications. Issues like data volatility (frequency of database updates), data currency (how important is an up-to-date representation), and data security and protection influence the decision of whether to access the "external" data through the DBMS, or whether that data can be duplicated, restructured, and permanently stored as part of the ES's own knowledge base.

A representative example is the case of PROBWELL, an ES for the detection of problems in oil wells [OLSO82], which requires data from a large operational database under IMS. Also, in an NYU project on Expert Systems for Business, it was determined that a life-insurance underwriting expert needs data from several external data sources (large customer databases, actuarial tables, etc. [JARK84a]).

It is mostly speculation whether other ESs reported in the literature might have profited from access to data stored in a DBMS. But since the mechanisms for such linkage were not available, this need may have been buried. It seems reasonable to suppose, however, that as ESs find wider commercial applications, we will see more interaction between ES and DBMS.

Determinism in Data Access Requirements. - Let us assume that the the database is large enough that only portions of it can reside in main storage at any one time. The knowledge of what particular information will be relevant to the

----

[3] Roughly, an entity in Internist corresponds to a tuple of a relation in a relational database, and a manifestation corresponds to an attribute.

decision-making process during an ES session is in some cases available in advance, while in other situations it can only be determined during the course of the user/system interaction. If the data requirements can be pre-determined, this directly implies the possibility of making "static" decisions, as to which portion of the database is actually needed by the ES. An example of where "static", pre-determined decisions are not feasible is the life-insurance underwriting expert at NYU. In each ES session, the customer profile as defined by the insurance application is completely different, thus requiring separate actuarial tables and related customer-account records. Furthermore, as each piece of data is examined in light of the evolving customer profile, additional data requirements are dynamically generated. In general, the use of "variables" as parameters in data access is a strong indicator of non-determinism in ESs.

4.0 STRATEGIES FOR DATABASE ACCESS IN EXPERT SYSTEMS

Four strategies for establishing a cooperative communication between the deductive and database access components of an expert system have been identified in [VASS83]. The spectrum of possible enhancements of an expert system with data management facilities is essentially a continuum. Starting from <u>elementary facilities for data retrieval</u>, e.g., a file system, we progress to a <u>generalized DBMS within the expert system</u>, to a <u>'loose' coupling</u> of the ES with an existing commercial DBMS, and finally to a <u>'tight' coupling</u> with an external DBMS. The four strategies can be considered to be subcases of two general architectures. The deductive and the database access components of an expert system can either be <u>integrated</u> into one system (the ES), or be <u>independent</u> systems which interact through a communications protocol [VASS84]. This is illustrated in Figure 3.

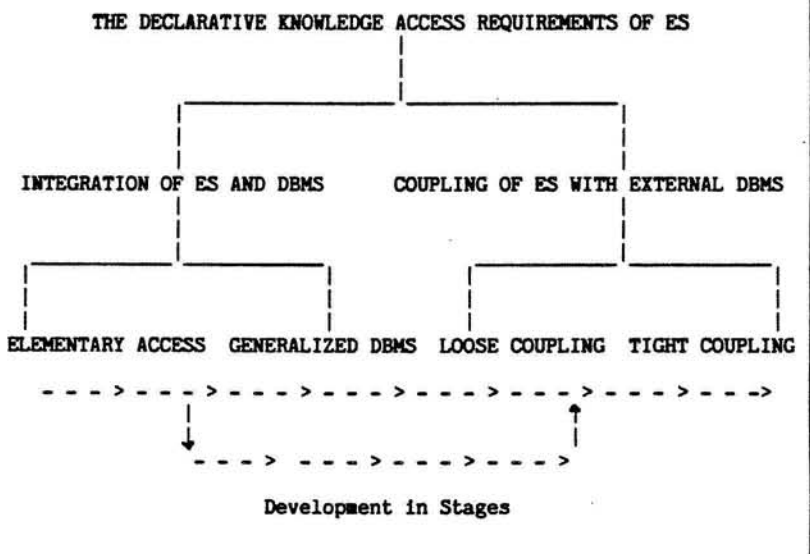

THE DECLARATIVE KNOWLEDGE ACCESS REQUIREMENTS OF ES
INTEGRATION OF ES AND DBMS
COUPLING OF ES WITH EXTERNAL DBMS
ELEMENTARY ACCESS
GENERALIZED DBMS
LOOSE COUPLING
TIGHT COUPLING
Development in Stages

Figure 3: STRATEGIES FOR DATABASE ACCESS BY AN ES

Expert system designers may choose one architecture over another depending on data volume, determinism in data access, and origin of the database, as described above; specific recommendations for the strategy selection problem are given in Section 4.5. These strategies could be developed independently, but in a careful design these successive enhancements would be incremental, allowing for a smooth transition to the next, more sophisticated strategy.

4.1 Elementary Database Access Within An Expert System - Strategy 1

The most commonly employed strategy for data access in ESs is to build an application-specific set of data structures and associated access routines in main memory. Systems like KRL, KLONE, FRL, NETL, and STROBE are used for the representation of the application-domain facts, but also provide data manipulation and simple access mechanisms. Other packages typically used in ESs exist in LISP libraries, e.g., DEFSTRUCT [MOON81], and The Record Package [MASI74]. Frequently, ES designers implement their own application-specific data-handling programs, as in the case of Internist [POPL77]. The direct use of the elementary database features of PROLOG (e.g. the "assert" predicate to add new facts, or the "retract" predicate to delete facts) is also an example of this Elementary Data Access Strategy.

What characterizes all such approaches is the direct manipulation of data objects in main storage. Otherwise, they differ in the degree of generality and data-independence they offer.

4.2 Generalized DBMS Within An Expert System - Strategy 2

In attempting to deal with large data volumes, efforts are underway to implement database management system components for an ES. For example, STROBE is being extended into a DBMS [LAFU83]. Because of its immediate correspondence with relational database concepts, PROLOG has been the favorite language for ES-internal DBMS implementations. For our life-insurance project we have implemented a generalized relational database management system within PROLOG to support a data dictionary with relation schemes, functional dependencies, external file handling, and the query capabilities of a relational algebra or calculus. This type of consolidation of database and programming language systems has also been investigated by [SCHM77]. PROLOG data access extensions to handle external file management are reported in [CHOM83] and [PARS83].

It is worth noting, however, that all these efforts are still early in development, and that they only mention as future plans the extensions of an Expert System to a full-pledged DBMS (i.e. accounting for data sharing, security, protection, etc.). Regardless, extending an ES to provide the facilities of a DBMS is a conceptually elegant approach, which may prove to have practical benefits in the long run.

4.3 Loose Coupling Of The ES With An External DBMS - Strategy 3

Conceptually the simplest solution to the problem of using existing databases managed by an external DBMS is to extract a snapshot of the required data from the DBMS when the ES begins to work on a set of related problems. This portion of the database is stored as the ES's own internal database in combination with any of the two previous access strategies. In current ESs, some form of loose coupling is used with Elementary Data Access mechanisms. Extensive use of this strategy, is limited by the degree of non-determinism in data access requirements, and by the fact that it incurs data-mapping overhead which for a large database can be quite high.

4.4 Tight Coupling Of The ES With An External DBMS - Strategy 4

In this access strategy the ES plays the role of an "intelligent" user of a generalized DBMS that manages a very large database. This is contrasted with loose coupling, where the ES is a one-time, simple user of the DBMS. Utilizing a tight-coupling scenario requires an online communication system between the ES and the DBMS. Queries can be generated and transmitted to the DBMS dynamically, and answers can be received and transformed into the internal knowledge representation. Thus in tight coupling the ES must know when and how to consult the DBMS, and must be able to understand the answers. The consequence of such dynamic use of the communication system is that the external database becomes an "extension" of the ES's knowledge base.

To attain tight coupling, particular care has to be taken to minimize the use of the communication system, and to the data representation and language translation problems. This is the topic of Section 5. To the authors' knowledge, tight coupling to an existing DBMS has not yet been implemented in actual systems, but research efforts are underway [KUNI82, VASS84]. It appears that the impact of logic programming and the commercialization of relational database systems will have a profound effect for tight coupling in future system architectures.

4.5 Recommendations In Choosing An Architecture

The designer of an expert system is faced with many architectural decisions, including how to structure the knowledge base, what form of inference engine is most appropriate, and so forth. We have discussed above the growing trend toward the development of ESs in environments involving very large databases; this introduces another decision for the ES designer, viz. what form of communication path between the ES and the database is appropriate?

Given the three dimensions along which we have analyzed the database access requirements of an ES, and the four strategies for ES-DBMS coupling, it becomes possible to provide some guidance for this decision by examining all of the combinations of system characteristics along these three dimensions, and suggesting appropriate coupling strategies for each. Figure 4 presents these suggestions in the form of a simple decision table. [4] For example, a situation involving very large data volume but no existing database is handled by Rule 3 in the table -- the suggestion for this situation is either to build into the ES generalized DBMS capabilities, or to acquire a DBMS to manage the large data volume, and use a tight-coupling strategy between the two systems. Other possible combinations of characteristics lead to different architectural recommendations.

As anyone who has developed an ES can readily attest, however, there are no really hard and fast rules that apply across the diverse domains to which ES technology has been and will continue to be applied. This table should therefore be viewed as a set of overall recommendations for broad categories of problem domains, rather than as a shortcut for avoiding careful examination of the unique characteristics of the problem at hand. For example, a recent paper [LAFU83] presents some empirical evidence that particular characteristics of the environment (both of the problem space itself and of the available hardware) often override an architectural decision made on purely theoretical grounds.

----

[4] The terminology used is the table is: At the condition part, N, Y denote NO and YES respectively and a dash "-" denotes that the entry could be either. At the action part, an X denotes that this action should be taken. It should be noted that, more than one X's in the same column (rule), imply that there is a choice of actions.

CONDITIONS	RULES 1	2	3	4	5
Is there a very large data volume?	N	N	Y	Y	Y
Is there an existing database?	-	-	N	Y	Y
Are the data access requirements determined during the session?	N	Y	-	N	Y
ACTIONS					
Strategy 1: Elementary Data Access	X	X			
Strategy 2: Extension of ES to Generalized DBMS		X	X		
Strategy 3: Loose Coupling with External DBMS	X			X	
Strategy 4: Tight Coupling with External DBMS			X		X

Figure 4: DECISION TABLE FOR THE DETERMINATION OF AN ES ARCHITECTURE

5.0 OPTIMIZATION REQUIREMENTS

The problem of optimizing database accesses by expert systems does not arise in all of the previously presented architectures. If strategy 1 is selected, the expert system will handle the database of specific facts in the same way as the general rules of the knowledge base. Strategy 2 (a DBMS within the expert system) permits the adaptation of any query processing strategies to the particular needs of the expert systems language. In strategy 3 (loose coupling), the expert system acts like any normal user of a DBMS and can be treated accordingly.

Optimization of the ES-DBMS interactions therefore becomes an issue only when a tight-coupling architecture (strategy 4) is employed. The way an expert system uses its data is typically very different from what a database management system is designed for. On the one hand, an expert system may issue a sequence of closely related calls, for instance, when using recursion in a logic-based representation. On the other hand, an expert system typically considers only one fact at a time whereas a relational DBMS can also work efficiently when confronted with set-oriented queries. Furthermore, this fact-at-a-time access requests will result in a very inefficient use of the communication system, as is demonstrated in [VASS84]. Since tight coupling has rarely been used to date in ES-DBMS, we shall limit the following discussion to a review of some recent work at NYU [VASS83, VASS84, JARK84b] for interfacing a Prolog-based expert system with a relational database system which supports an SQL interface.

During the conversion of Prolog predicates into SQL queries, the above-mentioned problems are solved by use of an intermediate language. This language mediates between Prolog and SQL in the sense that it is a (variable-free) subset of Prolog but - like SQL - gives a set-oriented description of the desired data in terms of stored base relations rather than views (as in the original Prolog version).

The optimization process can thus be divided into three steps [JARK84b]. First, a preprocessing mechanism collects ES requests for data while simulating the ES deduction process [VASS83]. As soon as a database-related request is encountered in the expert system, it stops its reasoning process and gives control to a higher, or meta-language, program, which simulates the continuation of the database-related reasoning to collect similar database requests. In effect, this mechanism delays the submission of individual tuple-oriented queries by converting them into more "optimizable" set-oriented ones, expressed in the intermediate language.

In the second step, the intermediate language expressions are optimized. Two techniques are employed: semantic query simplification, and common subexpression optimization in recursive calls. Query simplification (described in the introductory chapter by Jarke in this book) removes redundant subexpressions from a query, based on idempotency laws in connection with the detection of certain tautologies and contradictions in a query. Semantic query simplification employs integrity constraints such as value bounds for attributes, functional dependencies, and subset dependencies between primary and foreign keys in relations to detect more such simplifications than those visible from the original syntactic form of the query.

To optimize recursive calls, we exploit the case when the answer to each query step submitted to the database constitutes part of the input to subsequent queries. To avoid re-calculations, the result of each step in the recursion is kept as an intermediate result until the entire recursive query has been evaluated. The complete answer -- later used in the further reasoning process after the meta-language processing -- is composed as the union of the results at each intermediate step.

The optimized query is then translated to the DBMS query language and executed by the DBMS. The answer is loaded into the internal expert system database, or -- if that is too small -- into a temporary database relation. Garbage collection may be required to preserve the internal database as a rapidly accessible buffer in the presence of multiple unrelated database calls.

Only after all this has been completed does the meta-level evaluation stop and return control to the object-level of the expert system. The latter can now continue its reasoning with all the relevant data for the current search process provided in the internal database. If a new branch (backtracking from the start of the meta-level evaluation) of reasoning is considered, a new database call may be issued.

It should be noted at this point that this solution is not the only possible approach. Although not in the immediate context of expert systems, various compilation methods have been proposed for so-called deductive databases, i.e., systems in which a special-purpose expert system is superimposed upon a particular database system. Compilation is quite easy in the context of non-recursive reasoning [REIT78, GRAN81] but requires the presence of iteration constructs in the target language otherwise [HENS82]. In [JARK84a], an overview of such approaches is presented in the general context of database-expert systems interaction.

6.0 CONCLUDING REMARKS

The issue of Expert System interaction with Database Management Systems is emerging as an area for research, and one which can provide both practical and theoretical payoffs as the problems become identified and solutions are proposed. It can be expected that both areas will profit from this interaction. Expert systems will become more robust as they provide more of the facilities that are considered standard in commercial DBMSs: centralized storage and control of valuable data, efficient storage and access routines for large volumes of data, the ability to support multiple users, rollback and recovery strategies, and so forth. Database Management Systems will become more intelligent as they adopt more powerful knowledge representation schemes. and by means of the deductive capabilities of Expert Systems, they will provide much more sophisticated querying and decision-making capabilities than mere information management.

We have described a number of different strategies for interfacing these two types of systems, from one extreme of merging the two systems into one, to a dynamically controlled interaction between two independent systems at the other end of the spectrum. Criteria that are relevant to the determination of the appropriate strategy to adopt for a particular problem domain were identified and discussed. The reader should bear in mind that there are very few real systems in operation today that have addressed the problems discussed in this chapter. It is clear that as interactions between knowledge-based systems and database management systems become more widespread, more issues will be identified and additional solutions proposed.

References

[BOBR77]
Bobrow, D.G., and Winograd, T. "An Overview of KRL, a Knowledge Representation Language," Cognitive Science, Vol. 1, No. 1, 1977, pp. 84-123.

[BOWE82]
Bowen, K.A., and Kowalski, R.A., "Amalgamating Language and Metalanguage in Logic Programming," Logic Programming, K. Clark and S.A. Tarnlund, eds., Academic Press, 1982.

[BRAC79]
Brachman, R. "On the Epistemological Status of Semantic Networks," Associative Networks: Representation and Use of Knowledge by Computer, N.V. Findler, ed., Academic Press, 1979, pp. 3-50.

[BROD84]
Brodie, M. Mylopoulos, J., Schmidt, J.W. (eds.), On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages, Springer, Berlin-Heidelberg-New York, 1984.

[BROW81]
Brown, J., Burton, R, deKleer, J., "Pedagogical Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III," Intelligent Tutoring Systems, Sleeman et al (eds), Academic Press, 1981.

[CHOM83]
Chomicki, J., "A Database Support System for PROLOG," Proceedings of Logic Programming Workshop'83, Portugal, 1983.

[CLIF83]
Clifford, J., Jarke, M., and Y. Vassiliou, "A Short Introduction to Expert Systems," IEEE Database Engineering Bulletin, Vol.6, no.4, December 1983, pp.3-16.

[DAVI81]
Davis, R., "Expert Systems: Where are we? and Where do we Go from Here?," Massachusetts Institute of Technology, AI MEMO No. 665., June, 1982.

[ERMA80]
Erman, L.D., et al., "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," Computing Surveys, Vol. 12, No. 2, June 1980, pp. 213-253.

[FAHL79]
Fahlman, S.E. NETL: A System for Representing and Using Real-World Knowledge, MIT Press, Cambridge, Mass., 1979.

[FORG80]
Forgy, C.L. The OPS5 User's Manual, Tech. Report Carnegie-Mellon University, 1980.

[GALL78]
Gallaire, H., and Minker, J., Logic and Databases, Plenum, 1978.

[GEVA82]
Gevarter, William, B., "An Overview of Expert Systems," National Bureau of Standards Report, NBSIR 82-2505, 1982.

[GOLD77]
Goldstein, I., and S. Papert, "Artificial Intelligence, Language and the Study of Knowledge," Artificial Intelligence, Vol.1, No.1, 1977, pp.84-123.

[GRAN81]
Grant, J., Minker, J., "Optimization in deductive and conventional relational database systems," In Advances in Database Theory (Eds H. Gallaire, J. Minker, J.M. Nicolas), pp. 195-234. Plenum, New York.

[HART78]
Hart, P.E., Duda, R.O., and Einaudi, M.T. A Computer-Based Consultation System for Mineral Exploration, Tech. Report, SRI International, Menlo Park, Calif., 1978.

[HAYE83]
Hayes-Roth, F., Waterman, D.A., and D.B., Lenat, Building Expert Systems, Addison-Wesley, 1983.

[HENS82]
Henschen, L., Naqvi, S., "On compiling queries in recursive first-order databases," In Proceedings Workshop on Logical Bases for Data Bases. Toulouse.

[HEWI71]
Hewitt, C., "PLANNER: A Language for Proving Theorems in Robots," Proceedings IJCAI-71, London, England, 1971.

[JARK84a]
Jarke, M., and Y.Vassiliou, "Coupling Expert Systems with Database Management Systems," Artificial Intelligence Applications for Business (W.Reitman, ed.), Ablex, 1984, pp.65-85.

[JARK84b]
Jarke, M., Clifford, J., Vassiliou, Y., "An optimizing Prolog frontend to a relational query system," in Proceedings of ACM-SIGMOD Conference, Boston, June 1984.

[KOWA79]
Kowalski, R.A., Logic for Problem Solving. North-Holland, New York, 1979.

[KUNI83]
Kunifuji, S., Yokota, H., "Prolog and Relational Databases for Fifth Generation Computer Systems," Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

[LAFU83]
Lafue, G.M.E., "Basic decisions about linking an expert system with a DBMS: A case study," IEEE Database Engineering Bulletin Vol.6, No.4, December 1983.

[MASI74]
Masinter, L.M., in INTERLISP Reference Manual, 1974.

[MCDE81]
McDermott, J. "R1's Formative Years," AI Magazine Vol.2, No. 2, 1981.

[MINS75]
Minsky, M. "A Framework for Representing Knowledge," The Psychology of Computer Vision, P.H. Winston, ed., McGraw-Hill, New York, 1975, pp. 211-277.

[MOON81]
Moon, D., and D., Weinreb, in LISP Macine Manual, 1981.

[MYLO84]
Mylopoulos J., "An Overview of Knowledge Representation," in On Conceptual Modelling, (editors: M.L.Brodie, J.Mylopoulos, and J.W.Schmidt), Springer-Verlag, 1984, pp.3-18.

[NAU82]
Nau, D.S., "Expert Computer Systems," Computer, February, 1983, pp.63-85.

[NICO83]
Nicolas, J-M., and K. Yazdanian, "An Outline of BDGEN: A Deductive DBMS," Proceedings of Information Processing-83, 1983, pp.711-717.

[OLSO82]
Olson, J.P., and Ellis, S.P., "PROBWELL - An Expert Advisor for Determining Problems with Producing Wells," IBM Scientific/Engineering Conference, Poughkeepsie, New York, November, 1982.

[PARS83]
Parsaye, K., "Logic Programming and Relational Databases," IEEE Database Engineering Bulletin, vol.6, no.4, December 1983.

[PERE82]
Pereira, L.M., and Porto, A., "A Prolog Implementation of a Large System on a Small Machine," Departmento de Informatica, Universidade Nova de Lisboa, 1982.

[POPL77]
Pople, H., "The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning," Proceedings IJCAI-77, 1977, pp.1030-1037.

[POPL83]
Pople, H., "Knowledge Based Expert Systems: The Buy or Build Decision", Artificial Intelligence Applications for Business (W.Reitman, ed.), Ablex, January, 1984.

[REIT78]
Reiter, R., "Deductive question-answering on relational data bases," Logic and Databases (Eds H.Gallaire, J. Minker), pp. 149-178. Plenum, New York.

[SCHM77]
Schmidt, J., "Some High Level Language Constructs for Data of Type Relation," ACM Transaction on Database Systems, 2, 3, June, 1977, pp.247-261.

[SMIT83]
Smith, R.G., "STROBE: Support for Structured Object Knowledge Representation," in Proceedings of IJCAI-83, Karlsruhe, W.Germany, 1983.

[TSIC82]
Tsichritzis, D., and F.H., Lochovsky, Data Models, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[VANM79]
van Melle, W. "A Domain-Independent Production Rule System for Consultation Programs," Proc. Sixth Int'l Joint Conf. Artificial Intelligence, 1979.

[VASS83]
Vassiliou, Y., Clifford, J., Jarke, M., "How does an Expert System Get Its Data?," in Proc. 9th VLDB Conf., Florence, October 1983, pp.70-72.

[VASS84]
Vassiliou, Y., Clifford, J., Jarke, M., "Access to Specific Declarative Knowledge by Expert Systems," in Decision Support Systems, vol.1, no.1, to appear, 1984.

[WATE79]
Waterman, D., and F., Hayes-Roth, (eds), Pattern Directed Inference Systems, Academic Press, 1979.

[WONG77]
Wong, H.K.Y., and Mylopoulos, J., "Two Views on Data Semantics: Data Models in Artificial Intelligence and Database Management," INFOR, vol.15, no.3, 1977.