COUPLING EXPERT SYSTEMS WITH DATABASE MANAGEMENT SYSTEMS

Matthias Jarke

Yannis Vassiliou

May 1983

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

Working Paper Series

CRIS #54

GBA #83-53(CR)

ABSTRACT

The combined use of Database Management Systems (DBMS) and Artificial Intelligence-based Expert Systems (ES) is potentially very valuable for modern business applications. The large body of facts usually required in business information systems can be made available to an ES through an existing commercial DBMS. Furthermore, the DBMS itself can be used more intelligently and operated more efficiently if enhanced with ES features. However, the implementation of a DBMS-ES cooperation is very difficult.

We explore practical benefits of the cooperative use of DBMS and ES, as well as the research challenges it presents. Strategies for providing data from a DBMS to an ES are given; complementary strategies for providing intelligence from an ES to a DBMS are also presented. Finally, we discuss architectural issues such as degree of coupling, and combination with quantitative methods.

As an illustration, a research effort at New York University to integrate a logic-based business ES with a relational DBMS is described.

1.0 INTRODUCTION

In the mid-sixties the world watched with fascination a volcano off the fishing town of Heimaey, Iceland, create a new island, Surtsey, out of its ashes. Such things had happened before, and this particular creation process went on for four years. But, unlike on previous occasions, the new island did not vanish in the sea shortly after its creation but seemed determined to stay. Still, it is a rather tiny little island made visible to the world mainly by the big volcanic clouds that accompanied its creation.

Some people began to populate the new island and the problem arose how to provide them with the necessary things of life from the neighboring islands (Iceland being the most important). On the other hand, researchers from all over the world flocked in to unravel the hidden treasures of this new piece of land in the hope to clarify and resolve some of the problems in other areas. Currently, fishing boats manage the traffic but soon something more solid will be required to cope with the growing interaction.

What does all this have to do with expert systems and databases? The reader might have noticed a strong resemblance of Surtsey's development with the emergence of expert systems over the past few years. Once a small off shot of artificial intelligence, they now catch the spotlight of publicity after a prolonged period of laboratory existence.

Much research has been necessary to improve expert systems to commercial feasibility. With the actual intrusion of expert systems into the business world, however, the further question arises how to integrate them into the existing management information systems (MIS).

Can bridges (or at least fishing boats) be built to the other subsystems that supply the expert system with the necessary business data and allow them to use corporate planning models and other mathematical models as a human expert would?

Can, vice versa, the expert system idea be used to enhance existing MIS with more intelligence in systems design, usage, and operation?

Researchers who have addressed these problems typically looked at certain aspects of one of the two questions. In this paper, we try to view them together, with a primary focus on the interaction of expert systems with large existing databases. To remain in our original picture, we try to outline a bridge between the two systems that allows for two-way traffic enhancing the quality and efficiency of both types of systems.

The paper will analyze on a high level opportunities and solution strategies for getting business data from a commercial database system into an expert system, and for providing expert knowledge to database systems. For more technical details of the first problem, the reader is referred to a companion paper [Vassiliou et al. 1983] and to some work in the Japanese Fifth Generation Computer project [Kunifuji and Yokota 1982].

Certain aspects of the second problem are addressed in [Jarke and Koch 1983], [Jarke and Shalev 1983], as well as in the work of many others (e.g., [Reiter 1978], [Chang 1978], [Kellogg 1981], [King 1979], [Henschen et al. 1982]). However, the contribution of this paper is to unify all these approaches in a common architectural framework of what we believe might be the paradigm of forthcoming advanced decision support systems.

For an illustration, we will use from time to time examples from an ongoing project at New York University in which a logic-based business expert system is developed and interfaced with an existing relational database and mathematical subroutine libraries.

The paper is organized as follows. Section 2 gives a brief review of major subsystems and requirements in a business. In subsequent sections, we study the interaction between these components focusing on the relationships between databases and expert system knowledge bases and inference engines. Section 3 presents a stage-wise approach to the question how to get data for a business expert system. Section 4 deals with various applications of expert system technology to the design, use, and operation of database systems. Finally, section 5 addresses some architectural and technical problems of the coupling between expert and database management systems that have to be solved regardless of the direction of the interaction.

2.0 REVIEW OF BUSINESS SUBSYSTEMS AND REQUIREMENTS

In this section, the "islands" to be connected in a business will be identified, and their interaction requirements investigated. One type of current business systems is characterized by backbone transaction processing, governed by rules of what can be called a business program, and typically centered around large centralized or distributed databases. The knowledge required for building and working with such systems is typically hidden in procedural form and cannot be easily changed nor carried over to other similar applications. Modern transaction processing system types, such as office automation systems, make these rules more explicit and formalize the notion of documents as a central carrier of information to be handled [Tsichritzis 1982].

Another type of business systems deals more directly with supporting decisions on various levels. Decision support systems often have their knowledge built in mathematical formulas and models which are handled by model management algorithms or by the user via a flexible and powerful user interface [Stohr and White 1982].

Finally, there is a large number of human specialists who use experience and expertise together with factual knowledge gained from databases via query languages to develop recommendations and explanations in their area of expertise. Recently, expert systems try to efficiently support or partially replace such specialists. Expert systems typically organize their knowledge in the form of if-then rules and perform some form of pattern matching to find out which rules apply.

To summarize this discussion, it can be said that businesses use: (1) large databases, (2) mathematical formulas and models, (3) business rules and forms (documents), (4) experience and expertise of various human specialists. If expert systems are introduced, there will be another type of subsystem which interacts with the human decision-makers to complement their expertise.

In this kind of setting it is not hard to see that much effort is duplicated if all these systems operate independently of each other.

Today, most mathematical and expert subsystems have their own data management facilities. If data from a database system are used they are typically extracted as snapshots before the beginning of a session. This approach causes problems if the amount of data to be extracted is large or cannot easily be determined in advance. If the snapshots are kept over an extended period of time, there is a problem of keeping them consistent with the main database.

Database management systems usually offer some mathematical (e.g., aggregation) capabilities in their user interfaces and elementary rule-based techniques (integrity constraints) to check their operations. However, often the user would like more sophisticated retrieval facilities which perform reasoning or mathematical operations on the data before presenting them to the user. A higher level of semantic knowledge and deductive capabilities built into the database system would not only make it more user-friendly but also safer and more efficient to operate.

Finally, on a higher level, it can be observed that system development effort is duplicated on a large scale in the analysis and design of business information systems. This happens within but also among organizations when systems of the same type are developed again and again without taking explicit advantage of knowledge from previous experiences.

Some research has been done to integrate the separate subsystems. Decision support systems have to base their mathematical modelling capabilities on solid database management technology to become cost-effective. By analogy, it can be argued that many expert systems which use a large population of specific facts need a communication channel to the corporate databases. Going a step further, we argue that future decision support systems will have to integrate all three components: database, mathematical subsystem, and knowledge base.

As an example for such a system, consider a life insurance consulting expert system currently under development in our group. The system develops, recommends, and explains customized life insurance policies for a customer. The system will assist a sales agent by

(a) extracting customer information from the corporate database and from interviews to deduce the needs for life insurance coverage in different stages of the life of a customer;

(b) classifying the requirements in terms of basic actuarial types of insurance, and relating them to applicable actuarial models in the mathematical subsystem;

(c) computing a premium for the customized policy, reducing it by already existing coverage from previous insurance, and comparing the remainder to the premium paying capabilities of the customer;

(d) analyzing the feasibility of the proposed policy in terms of legal requirements and corporate objectives;

(e) interacting with the customer to come up with an acceptable policy that satisfies the customer's needs and has an affordable premium.

Currently, customized policies are feasible only for major group policies since there is a lack of actuarial experts to support a process as outlined above. Individual customers can essentially only choose among a limited number of pre-packaged policy combinations (insurance products). When operational, the expert system will thus relieve a bottleneck which seriously impedes service quality.

On the other hand, such a system needs:

(a) information about customers and actuarial table data from a database system;

(b) a mathematical subsystem to execute actuarial computations efficiently;

(c) an (extensible) knowledge-based expert subsystem to extract information, to classify insurance needs, to check time-varying legal and corporate constraints, and to explain problems and alternatives.

We are implementing such a system using the logic language Prolog, a relational database system, and a mathematical subroutine library. This project is meant to serve as a demonstration of a typical architecture we expect for upcoming knowledge-based business decision support systems.

In the remainder of this paper, we shall focus on the interaction of two of the subsystems discussed in this section, namely, knowledge-based expert systems and databases. First, we explore alternatives of extracting data from large and/or existing databases - getting supply to the inhabitants of the new island. Later on, we turn our attention the other way: how can existing subsystems, especially database management systems, be improved when expert system technology becomes available?

3.0 DATA FOR EXPERT SYSTEMS

In a rule-based expert system, the "inference-engine" uses a set of rules (the knowledge base) and a collection of specific facts (database) to simulate the behaviour of a human expert in a specialized problem domain.

For instance, the life insurance consulting expert system of Section 2 uses rules like:

```
IF   C is a customer of a certain age and
     when the customer dies a certain amount becomes payable
THEN the customer is said to have a whole life insurance benefit.
```

and a database containing data about actual customers, insurance and annuity benefits, mortality values, etc.

A database is represented in terms of two basic dimensions: variety and population. For example, in a logic-based representation the different logic predicates ("customer", "covered by", etc.) reflect the variety, and the instances of these predicates ("Smith is a customer", "Smith is covered by term annuity", etc.) refer to the population.

Most expert system databases exhibit a large variety of facts. In contrast, the population of facts in such databases is more variable; ranging from a small "laboratory" set to a very large collection of facts. Thus, expert system databases differ from traditional commercial databases in that they tend to be more "wide" and less "deep".

In [Vassiliou et al 1983] we examined the problem of expert system database representation and retrieval. We now present a summary of our results. We start with an outline of a four-stage approach in Section 3.1. Then, for an illustration we briefly describe in Section 3.2 a working implementation of this approach in the programming language Prolog.

3.1 Four-Stages Of Database Management Development

Four strategies for establishing a cooperative communication between the deductive and data components of an expert system have been identified. Starting from elementary facilities for data retrieval we progress to a generalized DBMS within the expert system, to a 'loose' coupling of the ES with an existing commercial DBMS, finally, to a 'tight' coupling with an external DBMS. Expert system designers may opt for one configuration over another depending on data volume, multiplicity of data use, data volatility characteristics (how frequently data is changed), or data protection and security requirements. Regardless, in a careful design these enhancements are incremental, allowing for a smooth transition from a less to a more sophisticated environment.

3.1.1 Stage 1: Elementary Data Management Within The ES -

In the simplest case, all data is kept in core and stored in mostly ad-hoc data structures. Application-specific routines for data retrieval and modification are implemented.

3.1.2 Stage 2: Generalized Data Management Within The ES -

When the expert system database is large enough not to fit in core, elementary data management is not sufficient. Techniques are needed for external file management (e.g. secondary indexes, data directory, etc.). These techniques should preferably be application independent, since a small change in application descriptions may require an altogether different mechanism.

Furthermore, depending on the multiplicity of database use and the extent of fact variety required for the ES, general purpose database management facilities may be needed. Such facilities include "views", or dynamic database windows, available in most modern relational DBMSs [SQL, INGRES, ORACLE].

Another facility is an integrated data dictionary that allows for queries about the database structure. In a nutshell, a generalized DBMS integrated in the expert system may be necessary to deal effectively with large databases.

This stage of enhancing database management facilities seems to be the norm for expert systems that deal with large databases, even though not all such systems exhibit the same level of sophistication.

3.1.3 Stage 3: Loose Coupling Of The ES With An Existing DBMS -

Often, the need for consulting existing very large databases arises. Such databases will normally be managed by a commercial DBMS. In a typical example, [Olson and Ellis 1982] report experience with PROBWELL, an expert system used to determine problems with oil wells. A very important source of information for such determination is a large database stored under the IMS database system. Unfortunately, this database cannot be made available to the ES in a timely manner.

Existing external databases are typically very large, highly volatile, and used by several applications. Costs of storing data and maintaining consistency may prohibit the duplication of such a database for the sake of the expert system alone.

Loose coupling of an ES with an existing DBMS refers to the presence of a communication channel between the two systems which allows for data extraction from the existing database, and subsequent storage of this "snapshot" as an expert system database. Data extractions occur statically before the actual operation of the ES.

We note that the availability of generalized database management facilities within the expert system may greatly facilitate this process. After the internal expert system database has been expanded with an external database snapshot, it can be accessed as in stage 2.

3.1.4 Stage 4: Tight Coupling Of An ES With An Existing DBMS -

The main disadvantage of loose coupling is the non-applicability in cases where automated dynamic decisions, as to which database portion is required, are necessary. During the same ES session, many different portions of the external database may be required at different times; the requirements may not be predictable. In addition, if the external database is continuously updated, for instance in databases for commodity trading or ticket reservation, the snapshots become rapidly obsolete.

Tight coupling of an ES with an external DBMS refers to the use of the communication channel between the two systems in such a way that the external database appears to the ES as an extension of its own. Clearly, the most important consideration for the implementation of tight coupling is the "intelligent" management of the communication channel - when and how to use the channel.

Our suggested strategy, which we describe in more detail in Section 3.2.5, assumes the existence of a high-level, sophisticated mechanism within the ES - an expert system itself - that collects ES requests for data while simulating the ES deduction process. This expert extracts and translates the necessary data. The facilities described in Stages 2 and 3 are necessary for the implementation of tight coupling.

This completes our general discussion of getting data into expert systems. In the following subsection, we describe an application of this stage-wise approach to the problem of data storage and retrieval for a Prolog-based expert system using relational database management systems. Most of the mechanisms described below have been implemented and tested in experimental Prolog programs presented in [Vassiliou et al. 1983].

3.2 Illustration - Data For A Prolog-based Expert System

3.2.1 Brief Introduction To Prolog -

Prolog is a programming language based on a subset of first-order logic, the Horn-clauses. Roughly, this amounts to dropping disjunction from logical consequents, and talking only about definite antecedent-consequent relationships. There are three basic statements in Prolog [Nau 1983]:

:- P.	means	P is a goal to be proved
P.	means	P is an assertion
P :- Q,R,S	means	Q and R and S imply P

A Prolog program is a sequence of clauses whose variables are considered to be universally quantified. A clause has both a declarative and a procedural interpretation. Thus,

P :- Q,R,S

can be read declaratively:

P is true if Q and R and S are true

or, procedurally:

To satisfy P first satisfy Q and R and S.

Since more than one clause may be needed to define a predicate (goal), there is a corresponding AND/OR graph for each predicate. The execution of a program involves a depth-first search with backtracking on these graphs, and uses the unification process based on the resolution principle [Robinson 1965].

A knowledge base can be represented in first-order logic if the formulas are suitably interpreted. Therefore, Prolog may be used for the knowledge representation. Furthermore, Prolog has the advantage that it already has a very powerful inference engine in place. The unification algorithm used in Prolog is more powerful than simple pattern matching algorithms common in production systems.

3.2.2 Elementary Data Management - Naive Use Of Prolog -

On the simplest level of database management facilities enhancements, stage 1, the whole population of facts can be represented directly as the expert system database. While this approach is feasible for any expert system, using Prolog can take us a step further. Since Prolog does not distinguish between data and programs, it can be used both as a (relational) data representation and as a query language.

Relational databases can be represented directly in Prolog as a listing of all instantiated predicates corresponding to relation tuples. For instance, consider a small portion of the insurance database:

```
customer(3163, smith, 1935, london)
customer(3154, jones, 1942, atlanta)
...
covered_by(3163, 'term insurance')
covered_by(3163, 'whole life annuity')
...
```

While general relational schemas are defined only implicitly by the predicate names, Prolog can be used as a powerful mechanism to define "generalized views" via additional rules. Views are used in DBMS to allow for more flexible data access. Prolog rules differ from traditional view mechanisms [Date 1982] in that with the use of variables they can accept parameters making them equivalent in this respect to the selector language construct proposed for database programming languages [Mall et al. 1983]. Consider the following examples:

```
covered_by_many( Custom_id) :- covered_by( Custom_id, Benefit1)
                               covered_by( Custom_id, Benefit2),
                               not( Benefit1 = Benefit2).

special_customer( C_id) :- not( covered_by_many( C_id))
                           or( customer( C_id,N1,Y1,london),
                              customer( C_id,N2,Y2,paris)).
```

The first of these Prolog statements can be read as: "A customer with customer id Custom_id (a variable) is covered by many benefits, if he/she is covered by at least two benefits which are different". Similarly, the second example reads: "A customer is special if he/she is not covered by many benefits and lives in Paris or London".

In addition to representing data and view definitions, Prolog can also directly represent queries about base data or views. A query is simply a goal:

```
:- covered_by_many(C_id).
```

which when executed will return in C_id the customer id of a customer who is covered by many insurance benefits.

3.2.3 Generalized Data Management - A DBMS Implemented In Prolog -

A further step towards integrating the deductive capabilities of Prolog with DBMS capabilities can be taken by implementing a general purpose DBMS directly in Prolog - stage 2 in our approach This can be done quite easily, and provides a means of adding flexible and general data access mechanisms to the inference engine without the need for a complicated interface to external database files.

In order to effect this stage in ES-DBMS coupling, the first requirement is the definition of an internal representation of a relational database. Given such a representation scheme, one can define any number of generalized operations to provide the facilities of a DBMS. Our approach [Vassiliou et al. 1983] provides a simple way to specify generalized relational operators acting on any relation and set of attributes. Prolog programs map from this simpler, user-oriented view of the operations to their implementation for the particular database and representation scheme chosen. This provides a degree of logical data independence as in the traditional levelled architecture of DBMSs [Date 1982].

A more application-specific solution is proposed in [Kunifuji and Yokota 1982]. [Kowalski 1981] details the use of Prolog for integrity constraints, database updates and historical databases.

Another advantage of a generalized DBMS within Prolog is efficiency. It is possible to devise a more sophisticated storage strategy (e.g., B-Trees), and perhaps to use auxiliary indexing schemes, hashing, etc. [Tarnlund 1978]. The work reported in [Pereira and Porto 1982] demonstrates that for specific applications, clever indexing schemes that guide decisions about which portions of external files should be read into the internal database can be devised. However, these strategies are not easily generalizable.

3.2.4 Loose Coupling Of A Prolog-based ES With A Relational DBMS -

Conceptually the simplest solution to the problem of using existing databases is to extract a snapshot of the required data from the DBMS before the ES begins to work on a set of related problems - stage 3 of the approach. The portion of the database is stored in the internal database of the ES as described previously with Prolog.

For this scenario to work, the following mechanisms are required:

1. Link to a DBMS with unload facilities;
2. Automatic generation of an ES database from the extracted database;
3. Knowing in advance which portion of the database is required for extraction (static decision).

3.2.5 Tight Coupling of a Prolog-based ES with a Relational DBMS

For the fourth and final stage of our approach as has been implemented with Prolog, we consider a very large existing database stored under a relational commercial DBMS. The naive use of the communication channel between the ES and the DBMS will assume the redirection of all ES queries, on predicates representing relations, to the DBMS for stored database relations. Any such approach is bound to face at least two major difficulties: the number of database calls will be prohibitevely many (each Prolog goal corresponds to a separate DBMS call), and the complexity of Prolog goals (queries) may make it impossible to translate them directly to DBMS queries (e.g. recursion).

These difficulties can be overcome by collecting and jointly executing database calls rather than executing them separately whenever required by the ES. In essence, the pure depth-first approach of Prolog is replaced by a combination of a depth-first reasoning and a breadth-first database call execution [Reiter 1978].

In practice, we use an amalgamation of the ES language with its own meta-language, based on the 'reflection principle' [Weyhrauch 1980]. This allows for a deferred evaluation of predicates requiring database calls, while at the same time the inference engine (theorem prover) of the ES is working.

Since all inferences are performed at the meta-level (simulation of object-level proofs), we are able to bring the complex ES queries into a form where some optimization and direct translation to a set of

DBMS queries is feasible. Note, that at this point it would be desirable to have an intelligent DBMS that collectively optimizes the execution of all the queries. We shall discuss such ideas in section 4.3, below.

The queries are directed to the DBMS, answers are obtained and transformed to the format accepted by the ES for internal databases. The ES can continue its reasoning at the object level. Each invocation of predicates corresponding to database relations will now amount to an ES database goal, rather than a call to an external DBMS.

By tight coupling, we are now able to use an existing large relational database as an extension of any internal Prolog database.

4.0 EXPERTS FOR DATABASE SYSTEMS

When compared to requirements for advanced business applications such as decision support for managerial users, current database management systems display a number of weaknesses. In this section, we highlight these problems and identify some strategies how they can be overcome by the use of rule-based techniques. We shall not go into detail about other artificial intelligence approaches such as natural language interfaces [Harris 1977, Wahlster 1981] or conceptual modelling techniques [Brodie and Zilles 1980, Brodie et al. 1983] which contribute to better database design and usage.

4.1 Problems With Current Database Systems

With the advent of decision support systems and the proliferation of computers and databases in general, the target user population for database query languages has changed: there are more (potential) users, such as managers and application specialists who have a high degree of application knowledge but little patience to acquire much familiarity with programming concepts [Jarke and Vassiliou 1982].

For such users, higher-level query languages are required which allow powerful operations without demanding major interaction skills. Besides the more ergonomic approach taken by the so-called second generation languages [Vassiliou and Jarke 1983], reasoning capabilities embedded in the system could allow for a wider range and more concise formulation of queries.

A second problem arises if users want fairly complex operations to be performed on the data which are not provided by the query language. Currently, they can only use a database programming language [Schmidt et al. 1982] instead of an ad-hoc query language, and program the operations on the data explicitly. This is not easy for non-computer specialists and often also prevents the use of DBMS report generation facilities burdening the user with even more programming tasks.

A third difficulty with most current DBMS is the lack of support for operations performed in a <u>context</u>. The system does not have a partner model of the user [Wahlster 1981]. Neither does it recognize that multiple users working on the system at the same time ask queries that could be answered collectively [Jarke et al. 1982]. Finally, a user who wishes to issue a query that is based on the result of a previous query has to store that result explicitly.

Besides not being as user-supportive as they could be, database systems could be improved towards more safety of the data and more efficiency of the execution of read and write transactions.

Safety is guaranteed in a database by enforcing integrity constraints, and by maintaining consistency through concurrency control. However, many DBMS support only rather simple types of integrity constraints because it is hard to test more complex combinations of constraints efficiently. Also, most systems do not support the resubmission of (sequences of) transactions after failure or user errors [Gray 1981].

Efficiency mainly refers to the response time for evaluating queries. Many systems do not recognize all special cases of query structure or content that permit the use of efficient special-purpose algorithms. Neither do they recognize sequences of queries where the output of one can be used as the input of the next (e.g., recursion, focusing, etc.).

In the next two subsections, we explore how rule-based technology can be used to improve on these problems. As in section 3, we take a stage-wise approach to the problem going from the user interface of the system towards its internal operations. It will be seen that many of the problems are related to each other and allow for common solution strategies.

Again as in section 3, the question arises whether these strategies can be used only in newly developed database systems, or whether an external expert system can be used via a bridge on top of an existing database system. We shall not address this question in this section but postpone it to section 5 where we discuss general issues common to both directions of interaction.

4.2 Expert Application 1: Intelligent Database Usage

In the past few years, much research has been directed towards making database interfaces more intelligent. The main thrust is towards more deductive capabilities while less is known about enhanced functions to be applied to retrieved data.

The first DBMS allowed access only to the stored files, records, and fields (relations, tuples, and attribute values in the relational model of data [Codd 1970]). More recent systems provide view mechanisms that allow the user to name windows through which only a subset of the database is visible. In the relational model, views are often called virtual relations. For example, a virtual relation "whole life insurance bearers" can be defined that contains customers who have a whole life insurance policy.

Traditional view mechanisms are somewhat limited. In the above example, a separate view of customers would have to be defined for each type of insurance benefit. The selector concept introduced in [Mall et al. 1982] allows the definition of views with parameters.

In our example, a selector

```
with_benefit (T) for customer_relation
```

can be defined where T can assume values such as "whole life insurance" or "term annuity". The user can now request or alter the value of a selected variable

```
customer[with_benefit("term annuity")]
```

which defines a moving window whose focus depends on the parameter chosen.

For querying (not updating) purposes, Prolog offers similar facilities in the "generalized view" rules presented in section 3. The Prolog inference engine can perform deductions on these view definitions, and thus answer queries not immediately answerable from the stored data.

A number of researchers have investigated these possibilities and some implemented systems exist. The solutions can be classified in much the same way as the stages given in section 3. There are: integrated systems containing both the ES and the DBMS [Minker 1978], [Warren 1981]; loosely coupled systems where the ES does all its work before the DBMS is called [Chang 1978], [Grishman 1978], [Reiter 1978], [Henschen and Naqvi 1982], [Fishman and Naqvi 1982]; and finally the tightly coupled system DADM [Kellogg 1982] that interleaves deduction in the ES with partial search in the DBMS.

Besides offering the user a deductive formal query language, such systems can also support other high-level user interfaces such as natural language [Sagalowicz 1977].

Other approaches which additionally address intelligent updating mechanisms will be described in section 4.3.2, below.

4.3 Expert Application 2: Intelligent Database Operation

The addition of general rules and inference mechanisms to a database system may improve the user interface, but it also presents a challenge to the database implementationresearcher. Not only is the goal to execute deductive processing efficiently but the new approach can be exploited as well to improve the execution of more traditional database operations. We investigate two areas: deduction-based query optimization methods, and deduction-based integrity checking for update transactions.

4.3.1 Query Optimization -

It is of foremost importance to a DBMS that it contains a query evaluation subsystem which identifies the fastest way to execute a submitted query. To this purpose, a query is usually standardized, simplified, and transformed in a way that makes the application of fast special-purpose algorithms feasible [Jarke and Koch 1982]. Simplification and standardization can be quite complex if difficult queries are asked.

A deductive component may use meta rules that guide the choice among the many applicable query transformation rules. [Grishman 1978] and [Reiter 1978] describe applications of deduction to the simplification step. [Warren 1981] implements the well-known query transformation heuristics of testing sharp restrictions first, and separating detachable subqueries in logic.

[Jarke and Koch 1983] develop a generalized heuristic called range nesting that combines both ideas. The deduction mechanism, however, is deterministic and contained in the query language compiler. An extension to this mechanism will use meta rules to generate alternative strategies and compare their evaluation costs based on knowledge about storage structures and database statistics.

Another transformation strategy that uses deduction makes more direct use of the general rules that define the database intension [Reiter 1978]. Semantic query processing [King 1979, 1981], [Hammer and Zdonik 1980] applies the integrity constraints of the database to simplify the execution of queries.

Assume, for example, that the ES contains a legal constraint that a minor may not get more than $3000 annuity income. If now the database is queried for customers with annuities of more than $3000 (or $5000, etc.), minors can be excluded from the search. Depending on the physical database structure, this may or may not speed up execution. Thus, this strategy also needs meta rules to guide the application of constraints.

Finally, the simultaneous optimization of multiple queries [Jarke et al. 1982] can be supported by an ES. One strategy is to remember selected query results to be used later [Finkelstein 1982], another to recognize common subexpressions in a batch of queries. A last area where expert knowledge is helpful is the complex problem of selecting physical storage structures and access paths [Paige 1982].

4.3.2 Transaction Management -

Changes in the database must obey the general laws defined by static (i.e., data value, referential) and dynamic (i.e., data value change) integrity constraints. The DBMS has to determine is which of the many general laws apply to a specific operation on specific data. Again, a pattern-matching oriented deduction process can be used [Nicolas and Yazdanian 1978], [Henschen et al. 1982]. Meta rules can be used to determine the possible delay of integrity checking until the changed data is actually needed [Lafue 1982]. While this idea may improve system efficiency it is unclear whether it is feasible from a management standpoint to keep the integrity of the database unknown over an extended period of time.

A second question also affects the user interface of the system: how to react to violations? [Nicolas and Yazdanian 1978] see three alternatives: reject the operation; accept the operation but do not change the database until further operations are submitted so that the transaction as a whole leaves the database in a consistent state [Gray 1981]; or trigger automatic changes to other data items to bring the database back to a consistent state.

The appropriate choice depends on many factors which can be partially controlled by meta rules and partially by the user. A knowledge-based database architecture proposed in [Jarke and Shalev 1983] contains an input management system to avoid unnecessary resubmission of transactions in such cases.

Language constructs such as the selector mechanism described in section 4.2 can be used for both query optimization and transaction control. First, the definition predicate of the selector can be used to identify the applicable integrity constraints for semantic query optimization and update control. Second, the selector defines candidate physical access paths which may lend themselves to "view indexing" [Roussopoulos 1982] in which the collection of answers to the parameterized selection predicate is stored as a physical access path. [Paige 1982] describes a method of "finite differencing" which dynamically generates and maintains views for query optimization and integrity control purposes.

5.0 ARCHITECTURAL AND TECHNICAL ISSUES IN COUPLING

To summarize our overview of ES applications to database management, it can be said that the techniques used both within and between the two proposed stages of intelligent use and intelligent operation are very similar. A system that incorporates most of the proposed strategies in a uniform architecture should be clearly feasible. On the other hand, such systems would need the same coupling mechanisms as the ones described in section 3 which in turn might greatly benefit from more intelligent databases. Therefore, in this last section of the paper we pose some architectural questions looking at the interaction between ES and DBMS from a higher perspective.

One might think that the natural architecture would be a uniform integrated system written in and usable through one language: either an extended database programming language [Schmidt et al. 1982] with deductive capabilities, or an ES language with general programming capabilities, such as Prolog [Kowalski 1981, Walker 1983].

However, not only has each of these languages its own idiosyncrasies making it awkward to use in certain subsystems (e.g., complex computations in Prolog), but also this architecture defeats the whole purpose of exploiting existing DBMS and quantitative methods. We therefore have to exclude this theoretically elegant alternative.

Three candidate architectures have been identified for the coupling of independent systems. This categorization is based on where processing takes place and how the interaction is controlled. We discuss each architecture in turn.

One architecture calls for a total distribution of processing and control. The two systems interact by exchanging messages, as in an "actor" approach [Hewitt 1976, Dhar 1983]. Each interaction assumes a master to slave relationship between the originator and the receiver of the message, but both systems are self-contained and can be operated independently. An advantage of this architecture is a large degree of application and system independence, allowing for transportability to other ES and DBMSs. How much each system each system has to know about the other's capabilities is an important consideration. The duplication of knowledge representations may introduce the dangers of redundancy, namely inconsistency and incompatibility.

At the other extreme, concentration of processing and control, a system integration can be envisioned. One of the two subsystems (ES or DBMS) may assume a more dominant role. This approach has naturally been followed by most researchers who focus on one direction of the interaction between ES and DBMS, for example [Chang 1978, Fordyce and Sullivan 1983, Kellogg 1982]. The architecture suggests a more variable distribution of labor (e.g. where query optimization is done) than the typically predetermined labor separation of distributed systems. There is much flexibility and potential in such an architecture, at the expense of transportability. Another difficulty with this solution is the integration of additional external subsystems.

Finally, in a third architecture, processing is distributed but control is now the responsibility of a separate subsystem, a supervisor program. In essence, the supervisor performs all the necessary steps for interfacing the ES with the DBMS (e.g., translations), and manages the interaction between them. This appears as a compromise architecture with the main advantage of allowing for a smoother interaction with other subsystems. Such subsystems include mathematical models and modules for knowledge acquisition, generation of alternative solutions [Reitman 1982], database design, and user interfaces. The challenging research question would be how to implement a supervisor that makes full use of the capabilities of the various subsystems without duplicating their features.

Within each architecture, means must be provided to translate knowledge representations and transaction requests between subsystems. As pointed out in section 3, the similarity between Prolog and the relational data model makes this task relatively easy. If other database models are used, however, intermediate translation and optimization steps become necessary. Depending on the architecture chosen, it must be decided which subsystem is responsible for this task.

6.0 CONCLUDING REMARKS

Several practical benefits from the cooperative use of expert systems with database management systems were identified. The overall goal of our work is an advanced system effectively supporting business decisions. Such a system would integrate database management, model management, and expert system technology within the same architecture.

At New York University, we are exploring several strategies for this goal. A stage-wise approach requiring simultaneous research on complementary topics as outlined in this paper is being designed and implemented.

Acknowledgments

The work reported in this paper is supported in part by a joint study with International Business Machines Corporation. Jim Clifford designed and implemented the internal relational database system in Prolog and co-developed the staged approach to getting data into the expert system. Taracad Sivasankaran did a major part of the initial design and preliminary implementation of the insurance expert system used as an example in this paper. We also thank the other project members, Hank Lucas, Ted Stohr, Norm White, and especially Walt Reitman for encouragement and many helpful discussions. Finally, we thank Sibylle Hentsch of Hamburg University for aquainting us with the intricacies of volcanoes off Iceland.

References

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

2. R.Brachman, "On the Epistemological Status of Semantic Networks", in N.V.Findler (ed.), Associative Networks: Representation and Use of Knowledge by Computer, Academic Press, 1977, 3-50.

3. M.Brodie, S.Zilles, Proc. Workshop on Data Abstraction, Databases, and Conceptual Modelling, SIGMOD Record 11, 2 (1980).

4. M.Brodie, J.Mylopoulos, J.W.Schmidt (eds.), Perspectives on Conceptual Modelling, Springer 1983.

5. B.Chandrasekaran, "Expert Systems: Matching Techniques to Tasks", NYU Symposium on Artificial Intelligence Applications for Business, New York, May 1983.

6. C.L.Chang, "DEDUCE 2: Further Investigations of Deduction in Relational Databases", in H.Gallaire, J.Minker (eds.), Logic and Databases, Plenum 1978, 201-236.

7. W.F.Clocksin, and C.S.Mellish, Programming in Prolog, Springer 1981.

8. E.F.Codd, "A Relational Model for Large Shared Data Bases", CACM, Vol.13, No.6, June 1970, 377-387.

9. V.Dhar, "Designing an Intelligent Decision Support System for Long Range Planning: An Artificial Intelligence Approach", Ph.D. Thesis Overview Report, Pittsburgh, February 1983.

10. R.Davis, "Expert Systems: Where are we? And where do we go from here?", Proc. 7th IJCAI, Vancouver, August 1981.

11. S.Finkelstein, "Common Expression Analysis in Database Applications", Proc. ACM-SIGMOD Conf., Orlando, Fl., 1982, 235-245.

12. D.Fishman, S.Naqvi, "An Intelligent Database System: AIDS", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

13. K.J.Fordyce, G.A.Sullivan, "Artificial Intelligence in Decision Support Systems: Presentation of an AI Technology and its Application in Working DSS", unpublished manuscript, Poughkeepsie, April 1983.

14. H.Gallaire, J.Minker (eds.), Logic and Databases, Plenum 1978.

15. J.Gray, "The Transaction Concept: Virtues and Limitations", Proc. 7th VLDB Conf., Cannes 1981, 144-154.

16. R.Grishman, "The Simplification of Retrieval Requests Generated by Question Answering Systems", Proc. 4th VLDB Conf., Berlin 1978, 400-406.

17. B.Grosz, "TEAM Extended Abstract", Proc. Philadelphia Database Interface Workshop (P.Buneman, ed.), Philadelphia, October 1982.

18. M.Hammer, S.Zdonik: "Knowledge-Based Query Processing", Proc. 6th VLDB Conf., Montreal 1980, 137-147.

19. L.R.Harris, "User-Oriented Data Base Query with the ROBOT Natural Language Query System", Proc. 3rd VLDB Conf., Tokyo 1977, 303-312.

20. L.Henschen, W.McCune, S.Naqvi, "Compiling Constraint-Checking Formulas from First-Order Formulas", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

21. L.Henschen, S.Naqvi, "On Compiling Queries in Recursive First-Order Databases", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

22. C.Hewitt, "Viewing Control Structures as Patterns of Passing Messages", AI Memo 410, MIT, Cambridge, December 1976.

23. M.Jarke, J.Koch, "A Survey of Query Optimization in Centralized Database Systems", NYU Working Paper Series, CRIS#44, GBA 82-73 (CR), November 1982.

24. M.Jarke, J.Koch, "Range Nesting - A Fast Method to Evaluate Quantified Queries", Proc. ACM-SIGMOD Conf., San Jose, May 1983.

25. M.Jarke, J.Koch, M.Mall, J.W.Schmidt, "Query Optimization Research in the Database Programming Languages (DBPL) Project", Database Engineering 5 (September 1982), 11-14.

26. M.Jarke, J. Shalev, "A Database Architecture for Supporting Business Transactions", NYU Working Paper Series CRIS #51, GBA 83-27 (CR), submitted for publication.

27. M.Jarke, Y.Vassiliou: "Choosing a Database Query Language", submitted for publication, November 1982.

28. C.Kellogg: "Knowledge Management: A Practical Amalgam of Knowledge and Data Base Technology", Proc. NCAI 1982.

29. J.J.King, "Exploring the Use of Domain Knowledge for Query Processing Efficiency", Technical Report STAN-CS-79-781, Stanford University, December 1979.

30. J.J.King, "QUIST: A System for Semantic Query Optimization in Relational Data Bases", Proc. 7th VLDB Conf., Cannes 1981, 510-517.

31. R.Kowalski, "Logic as a Database Language", unpublished manuscript, Imperial College, London, July 1981.

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

33. G.Lafue, "Logical Foundations for Delayed Integrity Checking", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

34. M.Mall, M.Reimer, J.W.Schmidt, "Data Selection, Sharing, and Access Control in a Relational Scenario", in M.Brodie, J.Mylopoulos, J.W.Schmidt (eds.), Perspectives on Conceptual Modelling, Springer 1983.

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

36. S.Naqvi, D.Fishman, L.Henschen, "An Improved Compiling Technique for First-Order Databases", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

37. D.Nau, "Expert Computer Systems", Computer, February 1983, 63-85.

38. J.-M.Nicolas, K.Yazdanian, "Integrity Checking in Deductive Databases", in H.Gallaire, J.Minker (eds.), Logic and Databases, Plenum 1978, 325-344.

39. J.-M.Nicolas, R.Demolombe, "On the Stability of Relational Queries", Proc. Workshop on Logical Bases for Data Bases, Toulouse, December 1982.

40. J.P.Olson, S.P.Ellis, "PROBWELL - An Expert Advisor for Determining Problems with Producing Wells", *Proc. IBM Scientific/ Engineering Conference*, Poughkeepsie/N.Y., November 1982, 95-101.

41. R.Paige, "Applications of Finite Differencing to Database Integrity Control and Query/Transaction Optimization", *Proc. Workshop on Logical Bases for Data Bases*, Toulouse, December 1982.

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

43. R.Reiter, "Deductive Question-Answering on Relational Data Bases", in H.Gallaire, J. Minker (eds.), *Logic and Databases*, Plenum 1978, 149-178.

44. W.Reitman, "Applying Artificial Intelligence to Decision Support: Where Do the Good Alternatives Come From?", in M.J.Ginzberg, W.Reitman, E.A.Stohr (eds.), *Decision Support Systems*, North Holland Publ. Co., 1982.

45. J.A.Robinson, "A Machine Oriented Logic Based on the Resolution Principle", *JACM*, 1965, Vol.1, No.4, 23-41.

46. N.Roussopoulos, "View Indexing in Relational Databases", *ACM-TODS* 7 (June 1982).

47. D.Sagalowicz, "IDA: An Intelligent Data Access System", *Proc. 3rd VLDB Conf.*, Tokyo 1977, 293-302.

48. J.W.Schmidt, M.Mall, J.Koch, M.Jarke, "Database Programming Languages", in P.Buneman (ed.), *Proc. Philadelphia Database Interface Workshop*, Philadelphia, October 1982.

49. E.A. Stohr, N.H. White, "User Interfaces for Decision Support Systems: An Overview", *International Journal of Policy Analysis and Information Systems* 6, 4 (1982), 393-423.

50. R.C.Schank, *Conceptual Information Processing*, North-Holland, New York, 1975.

51. S.A.Tarnlund, "Logical Basis for Data Bases", unpublished, 1978.

52. L.Travis, C.Kellogg, "Deductive Power in Knowledge Management Systems: Ideas and Experiments", *Proc. Workshop on Logical Bases for Data Bases*, Toulouse, December 1982.

53. D.Tsichritzis, "Form Management", *CACM* 25 (1982), 453-478.

54. Y. Vassiliou, J. Clifford, M. Jarke, "How does an Expert System Get Its Data?", NYU Working Paper CRIS #50 , GBA 83-26 (CR), submitted for publication.

55. Y.Vassiliou, M.Jarke, "Query Languages - A Taxonomy", in Y.Vassiliou (ed.), Human Factors and Interactive Computer Systems, Ablex, Norwood, N.J. 1983.

56. W.Wahlster, "Natural Language AI Systems: State of the Art and Research Perspective", in J.Siekmann (ed.), Proc. GWAI 81, Springer 1981.

57. A.Walker, "Data Bases, Expert Systems, and PROLOG", NYU Symposium on Artificial Intelligence Applications for Business", New York, May 1983.

58. D.H.D.Warren, L.M.Pereira, F.Pereira, "PROLOG - The Language and its Implementation Compared with LISP", Proc. Symp. on AI and Programming Languages, SIGPLAN Notices, Vol.12, No.8, 1977, 109-115.

59. D.H.D.Warren, "Efficient Processing of Interactive Relational Data Base Queries Expressed in Logic", Proc. 7th VLDB Conf., Cannes 1981, 272-282.

60. D.Waterman, F.Hayes-Roth (eds), Pattern Directed Inference Systems, Academic Press, 1979.

61. R.Weyhrauch, "Prolegomena to a Theory of Mechanical Formal Reasoning", Artificial Intelligence, Vol.13, 1980, 133-170.