Investigations Into Database Management System Support For Expert System Shells

Verlyn M Johnson
Summary - PhD Thesis
University of Minnesota
January 1993

Preface

This summary contains the Abstract, Introduction, and References sections from the thesis. The complete thesis is available from:

UMI Disseration Services
300 North Zeeb Road
Ann Arbor
Michigan 48106-1346

Abstract

Many expert system shells are available for developing production rule based expert system applications. However, it is difficult to rapidly change those applications to respond to changing business conditions. Each shell has its own production rule language and inferencing capabilities. It is unclear what information can be shared (reused). Use of main memory instead of a shared, common source for rules constrains the size of applications and can result in duplication. Maintenance is not immediately available to existing inference sessions and updates made by a session only affect that session.

This thesis approaches production rules and working storage as data that can be managed by enhanced database management systems (DBMSs). Five expert system shells are studied. A composite (canonical) production rule syntax is developed which provides knowledge engineers with a common language for production rules. It is mapped into an integrated data model for use by tool developers who wish to design common production rule storage databases and maintenance tools. Extensions to the data model allow expert system shell developers to reduce main memory constraints by using a DBMS to store and manage execution data. The analysis performed in building the data model reveals where translation, system enhancements, or standard definitions are required to share production rules.

Two DBMS enhancements are defined to facilitate management of production rule and execution data (but which also have other applications). Reflexive indexes enable a DBMS to incrementally maintain transitive closures (including multiple tables, duplicates, side paths, and accumulated values) as a database index. They simplify query formats, and eliminate the need for recursive processing during retrieval. One use is to accumulate rule premise evaluation values during inferencing. The inference locking protocol allows concurrent, dynamic access by those maintaining and executing control data. For example, it provides greater flexibility in maintaining production rules by allowing knowledge engineers to use multiple versions and notification to control how updates to production rules affect other maintenance and inference sessions. The protocol can also be used to extend production rule capabilities by allowing production rules to maintain production rules concurrently with other maintenance and inference sessions.

Introduction

Motivation

The use of expert systems in industry is rapidly expanding. A large number of expert system shells are available for use in developing expert systems applications. Typically, each shell has its own language (syntax and semantics) for developing expert system applications. Each shell stores production rules in its own way. This environment leads to inefficient use of the information encoded in each shell as observed in (SaWi90):

> ″As more and more knowledge based applications (such as expert systems) are developed, so the problem of duplication of knowledge will become apparent. For example, in an office where a number of different expert systems are used, any change to the legislation regarding tax or to the knowledge concerning the company and its operation might affect a number of different expert systems and might need to be encoded in the appropriate form for each different expert system.″

Often rules are compiled into a run-time (execution) format, such as a load module. The amount of information that can be encoded in an expert system application is limited because main memory is used for working-storage data. The problem is apparent in some existing expert system applications, and will become more important in the future. As noted in (FoMc86):

> ″Systems such as R1, which has over 4000 rules and typically deals at any given time with 500 objects, each of which may have up to 125 attributes, have begun to push the limits of conventional architectures.″

Later in (FoMc86):

> ″The approach traditionally taken is to increase the address space and store all data in virtual memory, letting the operating system worry about the rest. With the application of AI to problems such as factory management, the need to store and access large volumes of data outside of virtual memory is critical.″

Thus, the current environment for expert system system shells is one where a common representation and management scheme for production rules does not exist. Dynamic, concurrent access to production rules is not supported. Expert system applications are typically compiled and loaded into main memory for execution. The size of the applications must be limited so that main memory can be used to hold production rules and the working-storage data required for their execution.

A result of this environment is that it is difficult to rapidly change expert system applications in response to changing business conditions. An example based on the manufacture of electronic cards illustrates this problem. Suppose that a manufacturer produces electronic cards used in computers. Expert system applications are used to debug the cards as part of an automated manufacturing line. Families of electronic card types have similar characteristics. Individual card types within each family have the family characteristics, but also have some characteristics unique to the card type. Card types within each family may be developed over a period of several years. Since the best available technology is to be used when building the debugging expert system applications, different expert system shells will be used to develop applications dealing with the same card family.

A separate expert system application is used to aid in debugging each type of card. This reduces the amount of information required in the application to what is required for a single card type. However, the expert system application for each card type must include debugging information for both the card family

and the individual card type. Thus, each must include production rules which apply to the entire card family, such as a rule which invokes a common logic analyzer. For example:

```
R1 - IF   the card failed in the basic assurance test
     THEN check the oscillator,
          check the instruction number that was executing,
          execute hook-up instructions for logic analyzer '1'
```

Figure 1 illustrates the use of three different expert system shells for three card-debugging expert system applications containing a common rule.

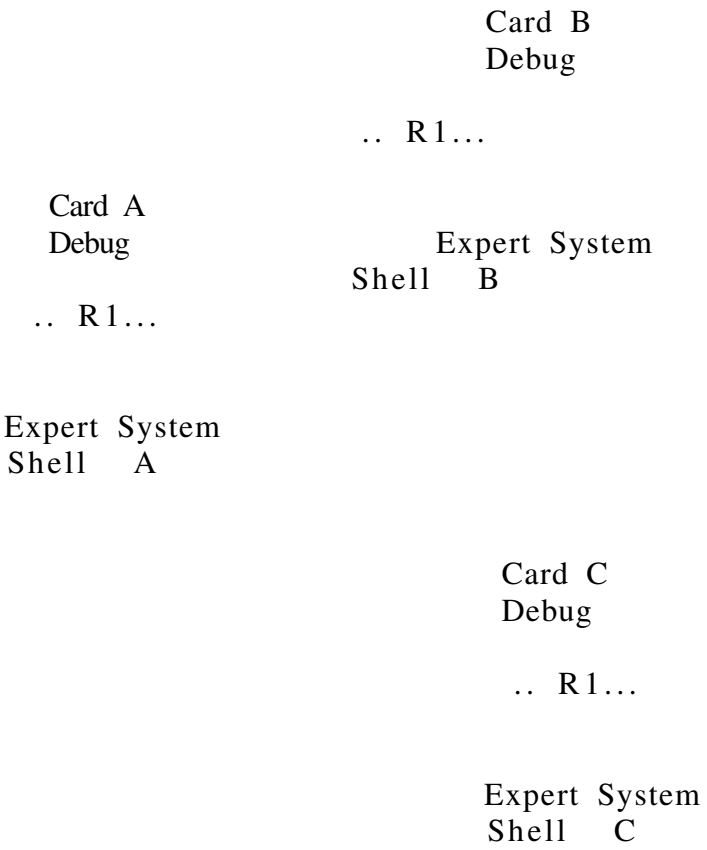

Figure 1. Use of three expert system shells for debugging cards in a family

Now, suppose the ′1′ analyzer becomes unavailable during the manufacturing day. This requires locating an alternative logic analyzer, or combination of logic analyzers which perform the required tests, and using them as a backup until the primary analyzer can be fixed. The original production rule could have included instructions for a backup analyzer if its identity were known. However, often the replacement, or set of replacements, is selected only at the time it is needed. It is not possible to predefine or table all of the potential

combinations. Suppose the production rule R1 must be modified to reflect the use of two backup logic analyzers:

```
R1 - IF   the card failed in the basic assurance test
     THEN check the oscillator
          check the instruction number that was executing
          execute hook-up instructions for 'backup' logic analyzer 2
          execute hook-up instructions for 'backup' logic analyzer 3
```

Three main problems in the current environment for expert system shells make it difficult to rapidly implement this change. They are:

1. The expert system applications which contain the production rule R1 were built using different expert system shells. These expert system shells do not share a common representation or source for the production rule R1.

2. The use of main memory to hold production rules during execution limits dynamic changes to production rules. The production rule R1 can not be updated directly.

3. The use of main memory to hold production rules and working-storage data during execution constrains the size of expert system applications. The production rule R1 must be repeated in several expert system applications.

The first problem that makes it difficult to rapidly change the production rule R1 is that several different expert system shells were used to build the debugging applications that contain the rule. Each of the expert system shells represents and manages the production rule R1 differently. Changing the rule in all of the debugging applications requires:

1. All of the expert system applications that contain the production rule must be located

2. The production rule must be updated for each of the expert system applications, using the language associated with the corresponding expert system shell

3. The new version of each expert system application must be released for use

Card debugging which requires logic analyzer processing will not take place properly until all of these steps are completed.

A question relating to this problem is, ″Why is the card manufacturer using three different expert system shells in the first place?″ There could be any number of reasons for this, but a common one is that each expert system shell provided some new technology that the card manufacturer wanted to exploit. For example, IBM™ [1] currently supports three expert system shells. Each provides features not found in the others. It is not easy to determine what production rule information is the same between the different expert system shells and what is unique to each. Nor is it clear how different expert system shells compare in their inferencing with production rules and use of working memory structures. If these were understood, knowledge acquisition tools could be developed to assist expert system application developers in tailoring data that was already present for a different expert system shell environment.

Likewise, producers of expert system shells need to determine what extensions are required to fully support and inference with production rules from other expert system shells. It would be best if producers of expert system shells evolved those shells to the use of new technologies. The use of new technology

[1] IBM is a trademark of the IBM Corporation.

should not require a new expert system shell. An evolution to new technology will also help to protect investments in existing expert system applications.

A second problem which slows implementation of the change to the production rule R1 is that most current expert system shells use main memory to hold production rules during execution. Dynamic changes to the production rules are often limited to an inference session changing its own main memory copy rather than the production rules themselves. Recovery in the event of a system failure is handled by each expert system shell. Changes to production rules made outside of the inference session cannot take effect until working copies of the production rules are updated and new inference sessions are started. For example, a recompile and release process may be required, followed by a new inference session of the expert system application.

This leads to another question related to the dynamic change of the production rule R1: ″When should the new production rule R1 take effect?″ With current expert system shells, the change would most likely take effect only when the expert system application, which included the production rule, was recompiled and/or a new inference session started. Thus, cards that are currently being debugged by an executing expert system application may be sent to an unavailable logic analyzer because those hook-up instructions were in effect at the time execution began. However, if a human expert were evaluating the cards it is very likely that the expert would begin to use the backup to the logic analyzer immediately. The expert would not continue to send cards to an unavailable analyzer just because that analyzer was the one available when the debug process started. A human expert would use the new hook-up instructions as soon as they were available. Cards currently being debugged, as well as new cards to be debugged, would be sent to the backup logic analyzer.

The third problem which makes it difficult to rapidly change the production rule R1 is also related to the use of main memory. Most expert system shells use main memory to hold the production rules and information in working memory during the execution of an expert system application. Thus size of the expert system applications is constrained. At times this makes it necessary to duplicate production rules in several expert system applications, rather than having only one source for those rules. Having multiple copies of production rules complicates the management of those rules. The use of main memory for working-storage data also constrains the amount of data an expert system application can work with at one time. Some envision that future expert system applications will consist of millions of production rules, with access to many millions of facts (Br88). These applications will require shared, dynamic access to very large numbers of production rules and large amounts of working memory. New methods will be required to support and manage this type of expert system application.

These problems are very similar to those encountered in many database applications (Fr90). Typically there is a large amount of data to be stored, managed, and used. This data may be concurrently updated and accessed by multiple users. Each user accesses only a portion of the data stored in a common database, but expects to access that data according to his/her own view of the data. The database management system manages the data and provides common services (such as recovery) to all users of the data.

The premise of this research is that expert system shells can use common services supplied by an enhanced database management system to access and manage production rule and working-storage data. The objectives are to:

- Allow sharing (reuse) of production rule information between expert system shells

- Allow dynamic, concurrent access to those maintaining and inferencing with production rules
- Reduce main memory constraints (which can result in duplication of production rules) in expert system applications

Background

The task of integrating and understanding the roles of expert systems and database management systems is currently an active research area. The problem is being addressed from several perspectives. These include enhancing expert system shells to allow access to data managed by database management systems, adding capability to database management systems to support the use of production rules, using a dictionary or rule management system to store and manage production rules, and exploration into the sharing of knowledge bases. These projects provide useful concepts to this research.

Expert System Shell Enhancements

Work has been done to develop enhancements to expert system shells which provide access to data stored in databases. Three architectures seem to be most common. A loosely coupled system allows a query to be made to the database, and moves the data returned into the working memory of the expert system shell (GrArLu89, BeShHu90, Ho90). No attempt is made to coordinate concurrent access to the data by other users. The copy will be out of date if the database is updated while the expert system application is executing. In effect the expert system takes a one-time snapshot of the data.

A tightly coupled system (StHe88) interacts with the database management system like other application programs. It uses transactions to lock data in the

database when it is read or updated. All users of the database have access to updates when locks are released, but some inference sessions may lock other users out of large parts of the database. The database management system coordinates concurrent access to the data (GrArLu89, BeShHu90).

A coupled system allows the expert system shell to access data under control of a database management system. A database access may return or update single tuples or sets of data. In coupled systems, separation is maintained between the expert system shell and the database management system (EcSa90, Ke90, RoYe90). Each is an independent component rather than having a single system. Figure 2 illustrates a coupled architecture for an expert system shell and a database management system.

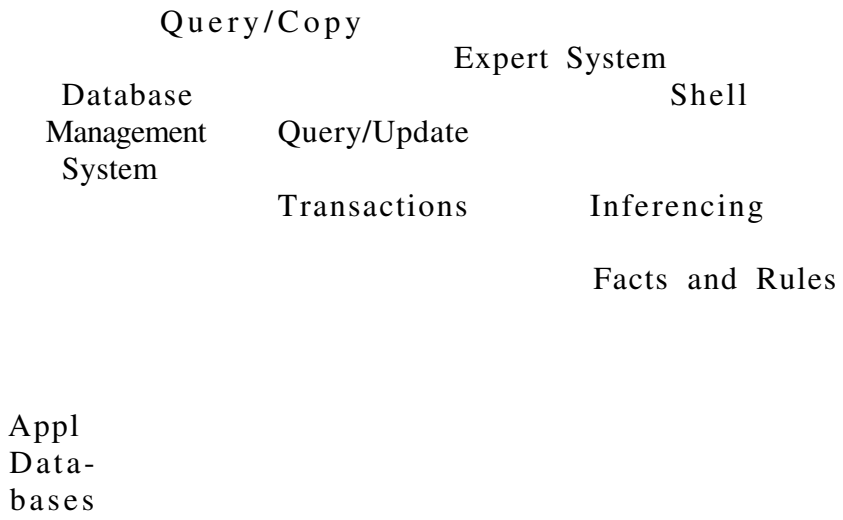

Figure 2. Coupled expert system shell and database management system

Blackboard architectures are another type of enhancement which has been explored. The architecture allows different expert system applications (or components) to communicate with each other by sharing information via a common global database (Ar88, Mu89, CoPa91). Each component uses information in the global database as input, and contributes additional information to the global database for use by other components. The components share a common understanding of the information in the global database. The database management system is used to manage the shared access to the global data. An example of this type of architecture is to share the state variables used in a simulation system (HaGr91).

None of these architectures use a database management system to manage the production rules as data. Nor do they examine the environment required to support multiple expert system shells. They are primarily targeted at allowing expert system applications to access data stored in databases. Thus, the amount of information available to the expert system is greater, and information can be shared between expert system applications, but the essential nature of the expert system shell is basically unchanged. The production rules themselves are not managed by the database management system. Main memory is still used for most working-storage data. The blackboard approach uses a database to allow concurrent access to global working data, but inferencing is still performed in main memory. This research views the production rules as data to be managed by the database management system. The database management system may also be used to manage working-storage data.

Knowledge Dictionary

Another method used to integrate expert systems and database management systems is to use a database or rule management system to manage the production rules, but to maintain separation between the database management

system and the expert system shell. This is the approach taken in the knowledge dictionary (JaCo88) and in rule management systems (Be-etal89, SaWi90). The advantage of this method is that many expert system shells can be supported by a single production rule storage source. It also allows the expert system shells to access various database management systems because the expert system shell language and inferencing are not physically bound to a single database management system. Maintaining physical separation of the expert system shell and database management system will simplify the task of accessing other sources of information when they become available.

Likewise, the KRISYS (Ma90) and KNOMAD (Ri90) projects are exploring the use of a DBMS to manage knowledge. Access can be either in a compiled or interpretive mode. A good database design can remove many performance problems, such as large amounts of data to read. Another group of projects are exploring the storage and execution of OPS5 production rules in relational databases (Se-etal87, SeLiRa88, IoSe89, SeLi90, SeLiRa90). This work has shown that OPS5 production rules and working-storage data can be efficiently stored and used for inference in relational tables. Other work has explored efficient inferencing with production rules in databases. Improved algorithms may be used (Mi87, He-etal90) or some prioritized sets of production rules can be transformed into a conflict free, priority free form (Ku91) to simplify conflict resolution.

Database programming languages (DeEt88, Sw89, KiMaSi90a, KiMaSi90b, Me91) also retain some separation between the inference session and the database management system. However, since each language is tailored to a specific database management system, the method is less general than the use of a knowledge dictionary or rule management system.

A knowledge dictionary provides the kinds of functions for expert system shells that data dictionaries provide for database management systems. Other functions are also provided such as:

- Documentation of the knowledge base
- Validation of the knowledge base
- Inference engine to process production rules stored as data
- Generation of a run-time format for production rules

The knowledge dictionary is based on an Entity-Relationship model for storing rules as data. Figure 3 illustrates a knowledge dictionary architecture.

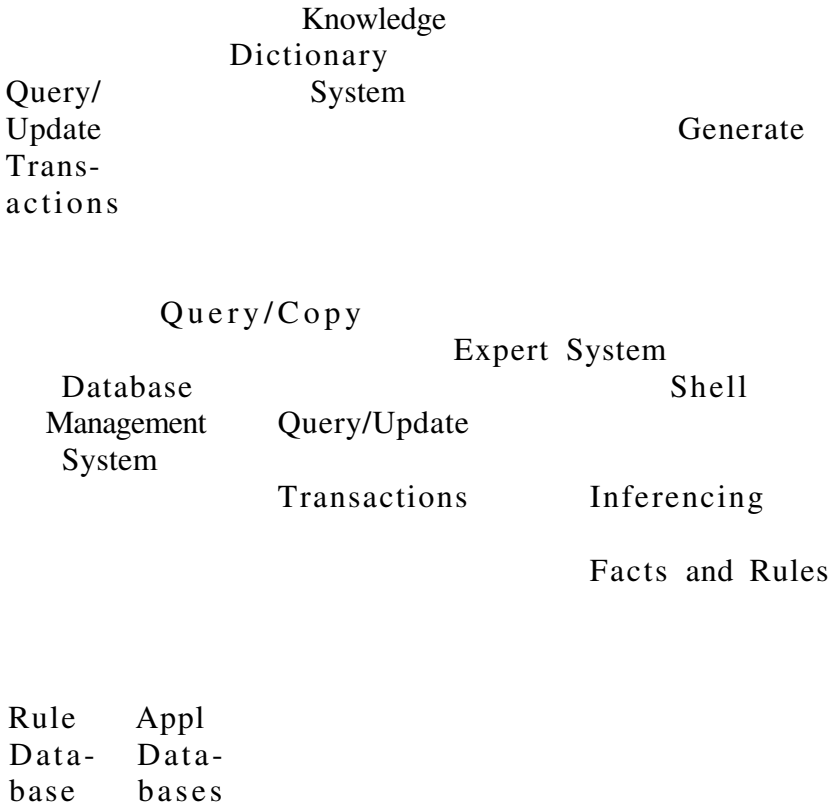

Figure 3. Knowledge dictionary architecture

One additional goal of the rule management system projects is to manage and support production rules from several expert system shells. However, instead of using a conventional database management system for the production rule data, a management system is developed ″from scratch″. Figure 4 illustrates a rule management system architecture.

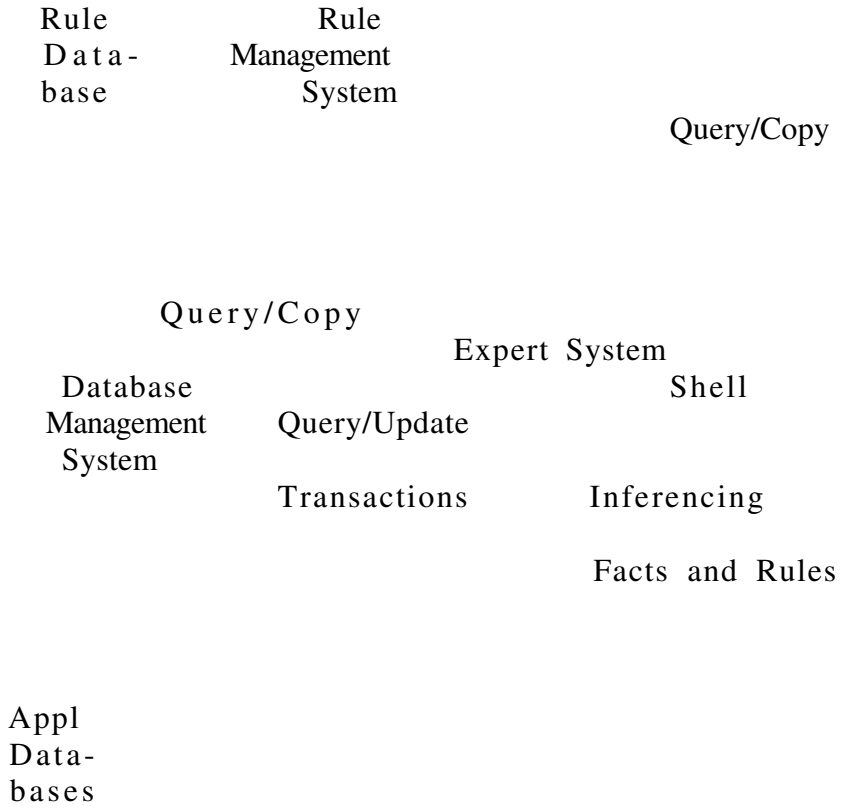

Figure 4. Rule management system architecture

Many of the stated objectives of the knowledge dictionary and rule management projects are similar to this research. The main differences are in scope and approach. The work with OPS5 production rules in relational databases and the knowledge dictionary Entity-Relationship model were not based on a detailed analysis of the production rules in several expert system shell languages. The knowledge dictionary authors identified generation of a run-time format for

production rules as a future enhancement, but will not be able to do this if the features used in the knowledge dictionary format are not supported in the run-time environment. Likewise, it will not be possible to store production rules from an existing system in the knowledge dictionary unless the Entity-Relationship model for the dictionary supports the features found in the source expert system shell language.

The work on the knowledge dictionary and rule management systems does not address enhancements to database management systems which facilitate the support of production rules. The knowledge dictionary project recognizes the need for this work and uses other research in the area (Ca86), but does not explore the area in their work. The KRISYS project explored the use of buffers to reduce access to the database. The work on rule management systems discusses some functions, but not as extensions to a database management system. This work uses a management system specifically for production rules.

Expert Database Systems

A large amount of research is taking place in the area of adding expert system-like functions to a database management system. These functions result in a system that can be used both to manage data and to build applications that reason about that data. Inferencing is included as an integral part of the database management system. This is often termed an expert database system or a deductive database system (Ke90,Za90,StHaPo88). These systems are tightly coupled because both the rules and the data are under control of the database management system. The metadatabase project has examined extending the concepts across heterogeneous, distributed environments (Hs-etal91).

In most systems two types of production rules may be defined. Data activated rules may be defined that are ′fired′ whenever the database management system

detects that the IF conditions are met. Chaining may occur if the actions on the THEN part of the rules result in the conditions being met for other rules. Inference rules allow inferencing to be performed as part of responding to query requests. Information can be derived that is not physically stored in the database. Figure 5 illustrates an expert database system architecture.

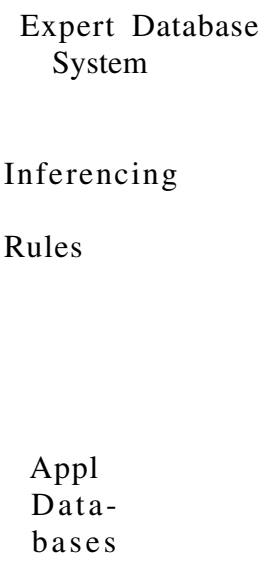

Figure 5. Expert database system architecture

The Postgres (StWoAn83, StHaHo87, RoSt87, StHaPo88, StHePo89) system supports the definition and storage of data-activated rules. The database management system (relational based) performs the action side of the rules when the specified conditions are met because of changes to the database. Examples of rule actions include database updates or actions to be taken if a constraint on a database is violated. A query request may also trigger rules which derive information from data stored in a database. This information may be returned along

with other data physically stored in the system. The Starburst system (WiFi89, WiCoLi91, AiWiHe91, CeWi92) is another project that is exploring similar concepts in a distributed environment. The HiPAC project (Ch89, McDa89, MyBr90) also uses active rules. However, this system uses both events and conditions as the basis for triggering rules. It concentrates more on the timing and scheduling aspects of the resulting database transactions and uses an object oriented architecture as its base.

The Logical Data Language (LDL) system (Ts88, Ch-etal90, KrZa88) is a good example of work in the area of deductive databases. It is an extension of the DATALOG language (La-etal90, GaKi90). LDL provides a database language based on Horn clause logic. The language is targeted at supporting complex queries and the development of complex applications. The LDL rules are compiled and optimized. This allows checks to be made for safety (queries will terminate) and allows the system to use the best performing execution method. Other methods used to enhance databases with logic are to extend PROLOG with database access (ChWa86, NuLiKo90, ZhHi90) or to extend the relational calculus with inference capability (McWe91).

The Relational Production Language (RPL) system (De88, DeEt89) is another example of an enhanced database language. RPL is based on an OPS5 production rule structure. It differs from some other systems in that it supports the use of a relationally structured area for working-storage data which resides in main memory. RPL applications are main memory resident, rather than using relational data bases for working memory. The RDL1 system (KiMaSi90a, KiMaSi90b) and RDL/C compiler (KiMa91) use a language that is similar to RPL. However, it includes support for programming constructs such as rule modules, abstract data types, and main memory variables. RDL1 is, in effect, a

rule-based programming language which provides integrated access to relational databases.

There are two main differences between this research and the above approaches. All of these systems developed their own language for database access. In many cases the languages were based on some existing production rule language, but no attempt was made to be able to support existing expert system applications in the new systems. This research explores a composite production rule syntax that can support existing applications. The composite production rule syntax includes the features supported by the production rule syntaxes of the expert system shell languages. The logical data model for storing the production rules includes attributes for specifying differences in how the expert system shells inference with the production rules.

The second difference with the above approaches (with the exception of Postgres) is that they require the rules to be compiled for run-time. The proposed research will result in a data model for storing rules so that either direct execution or generation to a complier can be performed. However, the emphasis will be on execution of expert system applications with support for concurrent access to the production rules by multiple users.

Database Management System Enhancements

An area of current research that is closely related to expert databases are enhancements to database management systems that will support the recursive queries found in expert database management systems. This work is important to using database management systems to support expert system shells because the structure of production rules is often reflexive. For example, expressions are reflexive in nature. The expression:

(A * (B - (D * E)))

has the structure:

E1 has term E2
E2 has term E3

where:

E1 is (A * (B - (D * E)))
E2 is (B - (D * E))
E3 is (D * E)

Or, in general :

Expression has term Expression

Thus, efficient retrieval of recursive data will be required when production rules are managed by a database management system.

Transitive closure is commonly used to satisfy recursive queries (BaRa86, Ro-etal86, MaSh90, Ed90a, Ed90b). This work describes extensions to the database query language to support recursive processing. The query statements can include information which allows the database management system to compute transitive closures as part of its processing to answer the queries. Some work has been done in the area of efficient algorithms. For example, (VaKh89) illustrates the parallel computation of an iterative transitive closure algorithm. The benefits of parallel processing may depend on the system characteristics and the base relation being processed (QaKi92). Materialized views (ToBl88, BlMa90, GuYu92) and surrogate files with hashed indexes (Ch90) have been

used to reduce the amount of database access required. At times updates can be computed from data already present and no additional access to the database is required. Rules may also be materialized (SeZh91), resulting in a trade-off between the speed gained and storage used. The best method depends on the depth of recursion and the size of the relations (AlRe91). Knowledge tables (Sh90) can greatly reduce the amount of space needed to represent the existence of specific relationships by representing them as a table of binary values. Others have looked at using various forms of indexes to speed the processing. The join index (Va87) prerecords the joining of two relations. This eliminates the need to perform the join as part of processing a query. In (Ja89) numerical values are used to allow compression of an index for a transitive closure. The goal is to be able to easily determine if two nodes in a network are connected. Unification has been discussed as one operation that would be useful as database management systems work with production rule data (YoIt86, YoKiHa89). This operation binds declarative knowledge instances to the production rules. Pattern matching is another operation that has been explored (HeCa87, TaSr92).

Another subject of current research that relates to the support of production rules by database management systems is examining enhancements to support the locking requirements in expert database systems. Locking will be required to support concurrent access by multiple users to a shared production rule database. Much of the research that has been done on locking has examined locking mechanisms for tuples covered by data-activated rules (StSeHa87, WiCoLi91) or interactions of multiple, possibly distributed production rules (AiWiHe92, CeWi92). The data is locked when the conditions for the rules are met. The locks allow for actions of one rule activating other rules. Some work has been done on locking of PROLOG-like rules in a concurrent environment (ChYa89). Rules are locked based on having the same predicate names and arguments.

The number of locks is controlled by using the concept of an existing lock 'covering' additional rules to be locked.

Other research on locking is addressing concurrency control for joint development environments and extending serial access to data. Many methods are explored, such as maintaining multiple copies (versions) of data (EiGi89), use of timestamps (KoSi88, LiTi88, ZhKa89), notification routines (Baka91), optimistic techniques (He90, RaTh90), predefining transaction overlaps (St-etal88), and compensating transactions (GaSa87). Some of these approaches do not concentrate on guaranteeing consistent data. Rather, they ensure conflicting transactions are aware of each other and have an opportunity to correct possible errors (BaKa91).

This research differs from previous work in two ways. First, it explores the design of an index which eliminates the need for recursive processing when retrieving data. The previous work reduces the time necessary to perform a transitive closure, but does not eliminate recursive processing for most queries. The intent is similar to the compressed transitive closure index, but the goal is to be able to retrieve data, not just indicate connectedness. Second, this work explores locking for production rules from several expert system shells, not just those in a PROLOG-like form. Previous locking discussions have not targeted the data access requirements for problems such as continuous operation without shutdown (Ed91) or machine-learning applications (Pa86, EmMo89, WhStLu90, AyKnSn91, DoKr91, Mo91). Locking must allow flexibility in the kind of locking protection that is requested and granted. At times it may be allowable to update rules currently involved in an inference session, and at other times updates should not be allowed. Conversion processing may be required when updates affect existing sessions. The previous work either assumes the same degree of locking (serializable) is desired at all times, the most recent version of

data should always be used, or does not allow for required conversion processing.

Knowledge Sharing / Standardization

Several other projects are exploring problems in sharing knowledge between expert system shells. Some projects are dealing with sharing information between expert system shells. The Initiative for Managing Knowledge Assets (IMKA) (CarnG90, Xe90, TaScYa91) is a consortium of several companies working to develop industry standard software for expert systems. They have developed an integrated representation for frame systems and intend to address production rules in the future. The Knowledge Interchange Format (KIF) (Ne-etal91, Gi91) and conceptual graphs (So92) have been proposed as predicate calculus based languages for exchanging knowledge. Another project has been proposed to develop an expert system language Hub (KnJa90). The Hub uses a 'Plain-Vanilla' production rule language for the expert system shells covered. The idea would be to translate knowledge bases from one language to another by first translating to the Hub language, and then from the Hub language to the target language. The authors recognize that the inferencing techniques of the rule language covered must be included in the Hub language. They also propose to write an inference engine that operates on the Hub language.

Some work has been done in the area of programming language translation (LeWu89, Co90). These projects illustrate that translating software depends on understanding the semantic operation, as well as the syntactic structure of the software. Another project (Li83) has explored storing programming language source as data. This work also examined accessing data from a compiled, executing program through database queries.

Having a standard structure for expert knowledge does not necessarily make it possible for different expert system applications to share that knowledge (Dee88, Ker90, SuPa90). It is also necessary to develop an integrated representation. This is similar to the need to develop integrated database designs, even if applications are using the same database management system. One project that is beginning to address the sharing of knowledge across expert system applications is the CYC project (Le89, Le-etal90, LeGu90). CYC is attempting to build a ′general knowledge′ base that can be accessed by many expert system shells. CYC is developing a management system for the information in its base, however in this case the method used to code the information is a frame system rather than production rules.

The work on IMKA, KIF, and conceptual graphs generally concentrates on representing declarative knowledge. It has not done an in-depth analysis of the production rules from different expert systems shells and how those production rules are used in inferencing. The proposed work on the ′Plain-Vanilla′ language is similar to developing the composite production rule syntax. However, the work does not use data modeling techniques to assist in developing the syntax, nor does it include storage of the production rules as data in a database management system. Both of these aspects will enhance the user′s ability to manage and manipulate the knowledge encoded in production rules. This research deals with these areas, and also explores the structure of the working-storage data that is required to inference with the production rules in order to gain a better understanding of the semantics associated with the various production rule languages. This work does not include the issue of developing an integrated view of knowledge for use by several expert system applications. It concentrates on the structure of production rules, not the structure of the knowledge represented as production rules.

Selected Expert System Shells

Many expert system shells use some form of production rules to represent and manipulate knowledge. This research examines a subset of these expert system shells. The expert system shells were chosen to reflect a sampling of the products and features available on the market today. They represent traditional approaches to production rule systems, as well as newer approaches, such as the use of production rules in a hybrid environment. Following is a brief description of each of the expert system shells examined in this research.

The Integrated Reasoning Shell

The Integrated Reasoning Shell (TIRS) is an AD/Cycle™ [2] tool which helps developers build knowledge-based applications (IBM90a, IBM90b). TIRS applications can be executed in various hardware and software environments. Graphics based tools aid in the development and testing of the application. Once the application is defined a build process is used to target the application to a specific run-time environment. Working-storage data is structured in frame systems or is stored as parameters. The production rules in TIRS specify the actions that are required to solve a problem. These include items such as updating working memory, executing external procedures, or reporting results.

Expert System Environment

Expert System Environment (ESE) is an expert system shell designed to aid non-dataprocessing professionals in building knowledge base applications (IBM88a, IBM88b). Tools exist to help users build screens, define the pro-

[2] AD/Cycle is a trademark of IBM Corporation.

duction rules, and control the portions of the knowledge base that are used. Parameters are used for internal storage of data. The production rules in the system provide the basic functions required to build simple matching systems. ESE attempts to make maintenance of expert system applications relatively easy.

KnowledgeTool

KnowledgeTool is an extension to PL/I that can be used to build knowledge based applications (IBM87). Data for the production rules is maintained as instances of classes. The classes are global and are declared outside the syntax of the production rules. Based on the instances in each class, bindings for each production rule are built and tracked by the inferencing system. Data may be passed back and forth between the KnowledgeTool portion of the code and the PL/I portion. The use of PL/I as a base allows a programmer to use all PL/I interfaces, such as access to database management systems and dialog manager tools.

Knowledge Engineering Environment

Intellicorp's® Knowledge Engineering Environment (KEE™) [3] is a hybird set of tools that can be used to build knowledge based applications (IntC86a, IntC86b, IntC86c, IntC86d, Tw88). Some functions supported include an object oriented approach to programming, truth maintenance, and graphics. Data in KEE is structured as frames. This can be extended to an object oriented approach by associating active value slots (methods) with the data structures. KEE is a LISP based system that supports full customization of system opera-

[3] IntelliCorp and KEE are trademarks of IntelliCorp.

tion. KEE also supports production rule-based reasoning on its internal data. Users may assert, or retract facts, or invoke inferencing to attempt to resolve queries.

Official Production System Version 5

Official Production System Version 5 (OPS5) is a production rule programming language that is commonly used to implement expert system applications (Br-etal85, CoWo88). It organizes data into classes and attributes like those in KnowledgeTool. In fact, KnowledgeTool is considered to be another implementation of the OPS5 use of production rules. The production rules match against instances of the data classes and take the appropriate actions when a match is found. Actions include items such as updating working memory, or outputting results. OPS5 also supports the definition of a new production rule as the action of another production rule.

Structure of the Investigation

This research views production rules and information in working memory as data which can be managed by an enhanced database management system. There are several areas to be investigated. They are:

- Sharing of production rule information by different expert system shells
- Similarities and differences in how different expert system shells inference with production rules
- Enhancements to database management systems which aid in the management of working-storage data and which allow dynamic, concurrent access to production rule data

- Use of production rules to build and maintain production rules in a dynamic, concurrent environment

Sharing of Production Rule Information

The first area investigated is the sharing (reuse) of production rule information between the selected expert system shells. To be able to share information it is necessary to understand what information is the same, what has some variation between production rule languages, and what is unique to a particular production rule language. Then, common representations of the information can be developed which support all of the features found in the production rule languages. Two viewpoints are explored. First, a user view (typically a knowledge engineer) of the production rule information is explored. A composite production rule syntax is developed which supports the production rules from each of the selected expert system shells. This means it is possible to translate production rules from each of the expert system shells into the composite production rule syntax. A Backus-Naur Form (BNF) is used to describe the composite production rule syntax. Chapter 2, ′Composite Production Rule Syntax′ describes the composite production rule syntax.

Second, a database management system view of the production rule information is examined. An integrated conceptual data model is developed which covers all of the production rule information. Individual production rule languages and expert system shells supply and use fragments of the integrated structure, depending on the features supported by each. The data model is built by mapping the composite production rule syntax into a conceptual data model. Appendix F, ′Expert System Shell Nonterminal/Entity Usage′, documents which nonterminals/entities are used to support production rules for each of the expert system shells. The data model will be a Logical Data Structure (LDS) that describes the data in production rules defined according to the composite pro-

duction rule syntax. The conceptual data model is analyzed to identify refinements that are required in the composite production rule syntax. As with any conceptual data model, it could also be used as the base for a database design for storing the production rule data. Examples of this are found in Appendix C, 'Sample KEE Expert System'; Appendix D, 'Sample TIRS Expert System'; and Appendix E, 'Sample ESE Expert System'. This will allow pieces of production rule data to be used by several production rules. These production rules may be from different expert system shells if both have equivalent syntactic structures in their production rule languages. Either production rule language might also require additional information that was not used in the other. An integrated database design will also allow common knowledge acquisition tools to be developed for maintaining the production rule data. Chapter 3, 'Composite Production Rule Data' describes how the conceptual data model for the composite production rule data was developed.

Inferencing Similarities and Differences

The second area investigated is the semantic similarities and differences in how the selected expert system shells inference with the production rules. This indicates what data is required by each of the expert systems shells during inferencing, how this data is used, and what features must be present in one expert system shell to support production rules developed for another expert system shell. Studying how the expert system shells inference with the production rules can also reveal refinements that are needed in the composite production rule syntax. The intent is to allow working-storage data to be managed by a database management system during inferencing. Extensions to the conceptual data model are defined for data that is required during inferencing by any of the expert system shells. Chapter 4, 'Composite Production Rule Execution Data' describes how the extensions to the conceptual data model were developed. Appendix F, 'Expert System Shell Nonterminal/Entity Usage', documents which

entities are used to support inferencing with production rules in each of the expert system shells. The extended data model remains integrated in that data that is required by more than one expert system shell appears only once in the data model. The extensions to the conceptual data model could be used as the base for a database design for working-storage data. Using a database for working-storage data will reduce main memory constraints on expert system applications and will allow the expert system shell to use services supplied by a database management system to maintain the data. Examples of a working-storage database design are found in Appendix C, ′Sample KEE Expert System′; Appendix D, ′Sample TIRS Expert System′; and Appendix E, ′Sample ESE Expert System′.

The trace examples do not guarantee an errorless composite production rule syntax and conceptual data model. Even translating and testing hundreds of expert system applications built using each of the expert system shells would not guarantee covering every aspect of each of the systems. However, the examples do illustrate that the composite production rule syntax and conceptual data model work in some cases, and illustrate how the composite production rule syntax and data model can be corrected if errors are discovered.

An additional result of the first two parts of the research is a summary that illustrates which features are found in each of the expert system shells. This chart can be used to help position additional expert system shells with respect to the composite production rule syntax, or can be used to indicate if it is possible to translate a set of production rules from one production rule language to another (Can a set of production rules be transported to a different expert system shell?). A translation is not possible if the set of production rules to be translated uses features not found in the target production rule syntax. Chapter 5, ′Feature Comparison′ describes the features found in each of the expert

system shells. Appendix G, 'Expert System Shell Feature Similarity' compares the similarities and differences in how features are supported in each of the expert system shells.

Database Management System Enhancements

The third area this research explores is the management of production rule and working-storage data by an enhanced database management system. This work does not try to identify and explore all useful enhancements. Rather, solutions to specific problems related to storing and managing production rule and working-storage data are investigated. The investigation is discussed in terms of relational database management systems. However, the types of functions required to support expert system shells have applications with other types of data, and the concepts could be implemented in other types of database management systems.

One characteristic of production rule data is that it is very reflexive. A prime example is an expression that consists of terms and operators. A term may itself be an expression. This same type of structure is common in other types of data as well. A classic example is a parent - child relationship. Chapter 6, 'Reflexive Index' explores enhancements to the indexes which can currently be defined on relational databases which would simplify and speed access to any data that is reflexive in nature. It then discusses additional enhancements and functions to meet the requirements of accessing production rule and working-storage data when inferencing in an expert system application.

A second characteristic of production rule data is that it is desirable to allow concurrent access to production rules between those inferencing with the production rules and those maintaining them. Locking in database management systems has typically taken the form of, "If a transaction has a write lock on a

unit, or an exclusive read lock on a unit, deny others access to it." There are two problems to this approach if the data to be locked is production rule data. First, the size of the units locked are determined by the database management system. For example, typical lock sizes are tuple, page, or table. This will not work well for production rules because of interdependencies between production rules in an expert system application. It is not possible to identify all of the interdependencies from the production rule data itself. For example, an external program executed as part of a rule action may set a global variable checked in another rule. There is no physical relationship between the two rules (such as a common predicate name) but a logical relationship exists.

Secondly, traditional read and write locks may not be flexible enough to support concurrently inferencing and updating a production rule database. For example, during inferencing, large groups of production rules are evaluated for matching conditions. Even those production rules whose premise evaluates to false may have to be protected from update because the updates may result in a logically inconsistent set of production rules. At times it is necessary to lock both production rules that are active in the inference chain and those that are not to ensure a correct execution of the expert system application. The duration of the locks must include multiple iterations through the recognize-act cycle.

On the other hand, it is not sufficient to simply lock all of the production rules in the expert system application for each inference session. This would prevent concurrent update access to the production rules by other users. The CARD_DEBUG manufacturing example illustrated that at times updates to production rules should take effect immediately (next iteration of the recognize-act cycle). In other cases, it is desirable to allow inference sessions in progress to complete using the old version of a production rule, but to have new sessions use the updated version.

Chapter 7, ′Inference Locking′ explores a locking protocol designed to allow concurrent access to those inferencing with and maintaining production rule data. It also explores application of the protocol to other types of data managed by a database management system.

Self-Modifying Production Rules

Some current expert system shells support a feature of production rules making limited changes to production rules. Typically, the scope of the changes is limited to the inference session which made the update. The locking protocol for production rule data allows extensions to be explored for this capability. Production rule actions are described which allow production rules to add, delete, or update production rules used by the current inference session or other inference sessions. The locking protocol also ensures that others who are making changes to the production rules are aware of inference session changes. Chapter 8, ′Self-Modifying Production Rules′ describes enhancements to the composite production rule syntax, storage data model, and execution data model to support this feature.

After the composite production rule syntax is verified extensions are explored which would effectively allow rule actions to create, delete, and modify production rules. Some AI languages (such as LISP and PROLOG) have this capability for working memory changes. If production rules are stored as data, and are managed by a database management system, concurrent use and update of the production rules by multiple users will be possible. It will be useful to allow expert system applications to make these changes, as well as knowledge engineers using knowledge acquisition tools.

Verification

Many types of problems allow a formal proof of the correctness of proposed solutions. Examples include whether or not a regular grammar is ambiguous (Gr71) or whether a locking protocol ensures serializable transactions (CyYa89). These types of proofs provide a firm theoretical base for future work. However, it is not possible to formally prove the correctness of proposed solutions to many types of interesting problems. For example, the general question of whether two context free languages are equal is undecidable (Gr71). This does not mean that it is impossible to find practical solutions to the problems or find evidence that proposed solutions are correct for many cases. It does mean that it is not possible to prove correctness for every possible case. The problems addressed by this thesis and their proposed solutions have this nature. They do not lend themselves to formal proofs of correctness. Thus, verification of the work has been done by using three other approaches.

First, the solutions have been verified by testing them against test case examples (refer to Appendix C, ′Sample KEE Expert System′; Appendix D, ′Sample TIRS Expert System′; and Appendix E, ′Sample ESE Expert System′). Production rules from three sample expert system applications have been translated to the composite production rule syntax. The conceptual data model for production rule data was verified by illustrating how the data for these production rules could be stored in relational tables mapped from the data model. The execution extensions to the conceptual data model have been verified by tracing the execution flow of the sample applications through relational tables mapped from the execution extensions. Examples have also been used to validate the operation of the reflexive index and inference locking protocol (refer to Chapter 6, ′Reflexive Index′ and Chapter 7, ′Inference Locking′). The use of test cases to verify the solutions is similar to debugging computer programs by testing their

execution. The test cases may not find every possible error, but the general structure and operation of the solution is verified.

Second, the composite production rule syntax and associated data models have been reviewed by several peers who have various degrees of expertise in the expert systems shells that were studied. The areas of expertise ranged from active researchers in the areas of database systems, those who are involved with architecting and developing some of the systems, to those who have used one or more of the systems to build expert system applications. These reviews were intended to serve the same purpose as the use of independent testers when validating computer programs. They were intended to:

- locate errors due to misunderstanding of how the studied systems function
- verify that the material is described in a way that is understandable
- locate errors which resulted from being overly familiar with the thesis (seeing what you mean/want to see in the document rather than what is there)

Third, and finally, in many cases latter work in the thesis has been based on earlier solutions which have been verified through the previous approaches. The conceptual data model is a direct mapping from the composite production rule syntax. The self-modifying production rule updates use the inference locking protocol and the execution extensions to the conceptual data model. Structuring the problems in this way assures that the latter work is consistent with the earlier solutions and also serves as another test case for the previous work.

It is clear that the above methods can not guarantee errorless solutions. Even translating and testing hundreds of expert system applications built using each of the expert system shells might not cover every possible aspect of each of the systems. However, the verification illustrates that the solutions work for some

cases, and provide some assurance that the general structure of the solutions are correct. Thus, if additional review or test cases reveal deficiencies, the verification that has been done should ensure that the solutions presented in the thesis can be modified or enhanced rather than having to search for whole new approaches to solving the problems. Finally, as is the case with computer programs, the use of the solutions for more and more cases will, over time, increase confidence that they are indeed ′correct′.

References

(AgBoJa89) Agrawal, R., Borgida, A., Jagadish, H.
"Efficient Management of Transitive Relationships in Large Data and Knowledge Bases."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 2, June, 1989.

(AiWiHe92) Aiken, A., Widom, J., Hellerstein, J.
"Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism."
IBM Research Report, January, 1992.

(AlRe91) Albert, L., Regnier, M.
"Complexity of Recursive Production Rules Execution."
3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems,
Rostock, May, 1991.

(Ar88) Arzen, K.
"An Architecture for Expert System-Based Feedback Control."
Artificial Intelligence in Real Time Control - Proceedings of the IFAC Workshop,
Swansea, September, 1988.

(AyKoSn91) Aytug, H., Koehler, G., Snowdon, J.
"Genetic Learning of Dynamic Scheduling within a Simulation Environment."
University of Florida and IBM - report in process,
November, 1991.

(BaKa90) Barghouti, N., Kaiser, G.
"Modeling Concurrency in Rule-Based Development Environments."
IEEE Expert, Volume 5, Number 6, December, 1990.

(BaKa91) Barghouti, N., Kaiser, G.
"Concurrency Control in Advanced Database Applications."
ACM Computing Surveys, Volume 23, Number 3, September, 1991.

(BaRa86) Bancihon, F., Ramakrishnan, R.
"An Amateur's Introduction to Recursive Query Processing Strategies."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 15, Number 2, June, 1986.

(BeDa88) Beauvieux, A., Dague, P.
"Interactive Checking of Knowledge Base Consistency."
Proceedings International Computer Conference - Artificial Intelligence: Theory and Applications,
Hong Kong, December, 1988.

(Be-etal89) Bert, M, et al.
"Rule Management for Heterogeneous Knowledge-based Systems."
CSELT Technical Reports, February, 1989.

(BeHaGo87) Berstein, P., Hadzilacos, V., Goodman, N.
Concurrency Control and Recovery in Database Systems.
Addison-Wesley Publishing Company, Inc.,
Reading, 1985.

(BeShHu90) Bell, D., Shao, J., Hull, M.
"Integrated Deductive Database System Implementation: A Systematic Study."
The Computer Journal, Volume 33, Number 1, 1990.

(BlMa90) Blakeley, J., Martin, N.
"Join Index, Materialized View, and Hybrid-Hash Join: A Performance Analysis."
Sixth International Conference on Data Engineering, Los Angeles, Feburary, 1990.

(Br88) Brodie, M.
"Integrating Databases and AI."
AAAI Conference (invited talk), St. Paul, 1988.

(Br-etal85) Brownston, L., et al.
Programming Expert Systems in OPS5
An Introduction to Rule-Based Programming.
Addison-Wesley Publishing Company, Inc., Reading, 1985.

(Ca86) Carlis, J.
"HAS, A Relational Algebra Operator, or Divide is not Enough to Conquer."
IEEE International Conference on Data Engineering, Los Angeles, 1986.

(Ca91) Carlis, J.
Logical Data Structures.
Computer Science Department, University of Minnesota, Minneapolis, 1991.

(CaMc90) Carpenter, W., McCrosky, W.
"A Multi-User, Relationally Based Infernece Engine."
Proceedings of the Fifth Annual AI Systems in Government Conference, Washington, D. C., May, 1990.

(CarnG90) Carnegie Group.
"Beta Release of Phase 1 IMKA Technology Announcement."
Press Release, November, 1990.

(CeWi92) Ceri, S., Widom, J.
"Production Rules in Parallel and Distributed Environments."
IBM Research Report, January, 1992.

(Ch89) Chakravarthy, S.
"Rule Management and Evaluation: An Active DBMS Perspective."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 3, September, 1989.

(Ch90) Chung, S.
"A Database Machine Based on Surrogate Files."
Proccedings of the 1990 International Conference on Application Specific Array Processors,
Piscataway, 1990.

(Ch-etal90) Chimenti D., et al.
"The LDL System Prototype."
IEEE Transactions on Knowledge and Data Engineering,
Volume 2, Number 1, March, 1990.

(ChWa86) Chang, C., Walker, A.
"PROSQL: A Prolog Programming Interface with SQL/DS."
Kerschberg, L. (ed).
Expert Data Base Systems--Proceedings From the First International Workshop.
The Benjamin/Cummings Publishing Company, Inc.,
Menlo Park, 1986.

(ChYa89) Chen, Y., Yang, W.
"A Mechanism for Concurrency Control in a Coupled Knowledge Base Management System."
Journal of Information Processing,
Volume 12, Number 4, 1989.

(Co90) Constales, D.
"Prototypes for the Automotic Translation of Computer Algebra Languages."
Proceedings, International Symposium DISCO '90, Capri, April, 1990.

(CoPa91) Collins, C., Parks, C.
"Manufacturing and Information Systems Issues: Software Architectures."
Computers & Industrial Engineering, Volume 21, Numbers 1-4, 1991.

(CoWo88) Cooper, T., Wogrin, N.
Rule-based Programming with OPS5
Morgan Kaufmann Publishers, Inc., San Mateo, 1988.

(Cs-etal91) Cser, L., et al.
"Three Kinds of Case-Based Learning in Sheet Metal Manufacturing."
Computers In Industry, Volume 17, Number 2, November, 1991.

(Da83) Date, C.
An Introduction to Database Systems - Volume II.
Addison-Wesley Publishing Company, Inc., Reading, 1985.

(Da86) Date, C.
An Introduction to Database Systems - Volume I.
Addison-Wesley Publishing Company, Inc., Reading, 1985.

(De88) Delcambre, L.
"RPL: An Expert System Language with Query Power."
IEEE Expert, Winter, 1988.

(Dee88) Deen, S.
"Research Issues in Federated Knowledge Based Systems."
Proceedings of Expert Systems 88, The Eigth Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems,
Brighton, December, 1988.

(DeEt88) Delcambre, L., Etheredge, J.
"A Self-Controlling Interpreter for the Relational Production Language."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 17, Number 3, September, 1988.

(DeEt89) Delcambre, L., Etheredge, J.
"The Relational Production Language: A Production Language for Relational Databases."
Kerschberg, L. (ed).
Proceedings from the Second International Conference on Expert Database Systems.
The Benjamin/Cummings Publishing Company, Inc.,
Redwood City, 1989.

(DoKr91) Dolk, D., Kridel, D.
"An Active Modeling System for Econometric Analysis."
Decision Support Systems, Volume 7, Number 4,
November, 1991.

(EcSa90) Eccles, J, Saldanha, J.
"Metadata-based Generation and Management of Knowledgebases from Molecular Biological Databases."
Computer Methods and Programs in Biomedicine,
June, 1990.

(Ed90a) Eder, J.
"General Transitive Closure of Relations Containing Duplicates."
Information Systems, Volume 15, Number 3, 1989.

(Ed90b) Eder, J.
"Extending SQL with General Transitive Closure and Extreme Value Selections."
IEEE Transactions on Knowledge and Data Engineering, Volume 2, Number 4, December, 1990.

(Ed91) Edelstein, H.
"Database World Targets Next-Generation Problems."
Software Magazine, May 1991.

(EiGi89) Ellis, C., Gibbs, S.
"Concurrency Control in Groupware Systems."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 2, June, 1989.

(EmMo89) Emde, W., Morik, K.
"Consultation Independent Learning in BLIP."
Machine and Human Learning. Advances in Eurpoean Research, 1989.

(En87) Engles, R.
"Structured Tables."
ANSI X3H2-87-331, 1987.

(FoMc86) Fox, M., McDermott, J.
"The Role of Databases in Knowledge-Based Systems."
Brodie M, Mylopoulos, J. (eds).
On Knowledge Base Management Systems.
Springer-Verlag, New York, 1986.

(Fr90) Freundlich, Y.
"Knowledge Bases and Databases - Converging Technologies, Diverging Interests."
Computer,
Volume 23, Number 11, November, 1990.

(GaKi90) Gardarin, G., Kiernan, J.
"Deductive Database Rule Languages: Analysis and Case Study."
Database Systems of the 90s. International Symposium Proceedings,
Berlin, November, 1990.

(GaSa87) Garcia-Molina, H., Salem, K.
"SAGAS."
Proceedings of Association For Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference,
San Francisco, May, 1987.

(Gi88) Ginsberg, A.
"Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy."
Proceedings of AAAI 88 - Seventh National Conference on Artificial Intelligence, St Paul, August, 1988.

(Gi91) Ginsberg, M.
"Knowledge Interchange Format: The KIF of Death."
AI Magazine, Volume 12, Number 3, 1991.

(Gr71) Gries, D.
Compilier Construction For Digital Computers.
John Wiley & Sons, Inc, New York, 1971.

(GrArLu89) Gray, P., Archibald I., Lunn K.
"Interfacing a Knowledge-based System to a Large Database."
The Knowledge Engineering Review,
Volume 4, Number 1, 1989.

(GuYu92) Guh, K., Yu, C.
"Efficient Management of Materialized Generalized Transitive Closure in Centralized and Parallel Environments."
IEEE Transactions on Knowledge and Data Engineering, Volume 4, Number 4, August, 1992.

(Ha-etal90) Hanson, E., et al.
"A Predicate Matching Algorithm for Database Rule Systems."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 19, Number 2, June, 1990.

(HaDo89) Halici, U., Dogac, A.
"Concurrency Control in Distributed Databases Through Time Intervals and Short Term Locks."
IEEE Transactions on Software Engineering,
Volume 15, Number 8, August, 1989.

(HaGr91) Hakman, M., Groth, T.
"KBSIM: A System for Interactive Knowledge-based Simulation."
Computer Methods and Programs in Biomedicine,
Volume 34, Numbers 2 and 3, February, March, 1991.

(He90) Herlihy, M.
"Apologizing Versus Asking Permission: Optmistic Concurrency Control for Abstract Data Types."
ACM Transactions on Database Systems,
Volume 15, Number 1, March, 1990.

(HeCa87) Held, J., Carlis, J.
"MATCH - A New High-Level Relational Operator For Pattern Matching."
Communications of the ACM,
Volume 30, Number 1, January, 1987.

(HeCa90) Held, J., Carlis, J.
"A Shared Conceptual Schema for Four Medical Expert Systems."
Sixth IEEE International Conference on Data Engineering, Los Angeles, 1990.

(Ho90) Hoevenaars, M.
"Integrating Expert Systems and Relational Databases."
Expert Systems Integration. Proceedings of the BANKAI Workshop, Brussels, September, 1990.

(HoJo89) Hoffman, J., Johnson, V.
"A Design for Indexes to Support Reflexive Relations."
IBM Technical Report 07.1118, October, 1989.

(HoLe87) Hong, B., Lee, S.
"Modeling of Version Relationships for CAD Databases."
Proceedings of TENCON87: 1987 Region 10 Conference, 'Computers and Communications Technology Toward 2000', Volume 1, Seoul, August, 1987.

(Hs-etal91) Hsu, C., et al.
"Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach."
IEEE Transactions on Software Engineering, Volume 17, Number 6, June, 1991.

(IBM86a) IBM Corporation.
IMS Application Development Facility II Version 2 Release 2 - Application Development Guide.
IBM product documentation (SH20-6595-01), 1986.

(IBM86b) IBM Corporation.
IMS Application Development Facility II Version 2 Release 2 - Application Development Reference.
IBM product documentation (SH20-6594-01), 1986.

(IBM87) IBM Corporation.
IBM KnowledgeTool - User's Guide and Reference Release 1.
IBM product documentation (SH20-9251-00), 1987.

(IBM88a) IBM Corporation.
Expert System Environment - Application Programming Guide.
IBM product documentation (SC38-7020-02), 1988.

(IBM88b) IBM Corporation.
Expert System Environment Reference Manual.
IBM product documentation (SC38-7004-1), 1988.

(IBM90a) IBM Corporation.
AD/Cycle - The Integrated Reasoning Shell Development/2 Application Design and Development Guide Version 1 Release 1.
IBM product documentation (SH21-1006-0), 1990.

(IBM90b) IBM Corporation.
The Integrated Reasoning Shell - Reference Manual Version 1 Release 1.
IBM product documentation (SH21-1007-0), 1990.

(IBM90c) IBM Corporation.
Repository Manager/MVS Supplied Entity-Relationshi p Model Definition Version 1 Release 1.
IBM product documentation (SC26-4830-00), 1990.

(IBM91a) IBM Corporation.
IBM Database 2 Version 2 Command and Utility Reference Release 3.
IBM product documentation (ZC26-4879-00), 1991.

(IBM91b) IBM Corporation.
IBM Database 2 Version 2 SQL Reference Release 3.
IBM product documentation (ZC26-4884-00), 1991.

(IntC86a) IntelliCorp.
IntelliCorp KEE - Software Development System 3.0 Training Supplementary Materials.
IntelliCorp product documentation (3.0-TSM-1), 1986.

(IntC86b) IntelliCorp.
IntelliCorp KEE - Education class handouts,
IntelliCorp product documentation, 1986.

(IntC86c) IntelliCorp.
IntelliCorp KEE - Software Development System 3.0 Training Manual (3.0-TZ-1).
IntelliCorp product documentation (3.0-TZ-1), 1986.

(IntC86d) IntelliCorp.
IntelliCorp KEE - Software Development System Rulesystem3 Reference Manual
IntelliCorp product documentation (3.0-RR-1), 1986.

(IoSe89) Ioannidis, Y., Sellis, T.
"Conflict Resolution of Rules Assigning Values to Virtual Attributes."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 2, June, 1989.

(Ja89) Jagadish, H.
"A Compressed Transitive Closure Technique for Effective Fixed-Point Query Processing."
Kerschberg, L (ed).
Proceedings from the Second International Conference on Expert Database Systems.
The Benjamin/Cummings Publishing Company, Inc.,
Redwood City, 1989.

(Ja90) Jagadish, H.
"A Compression Technique to Materialize Transitive Closure."
ACM Transactions on Database Systems,
December, 1990.

(JaCo88) Jansen, B., Compton, P.
"The Knowledge Dictionary - A Relational Tool for the Maintenance of Expert Systems."
Technical Report TR-FC-88-01,
Commonwealth Scientific and Industrial Research Organization,
North Ryde, Australia, February, 1988.

(JoCa90) Johnson, V., Carlis, J
"Design of a Composite Production Rule Syntax and Associated Logical Data Structure."
Summary-IBM Technical Disclosure Bulletin,
December, 1990.

(JoCa91) Johnson, V, Carlis, J,
"Composite Rule System. Investigations Into Using Database Management Systems In Support of Expert System Shells."
Proceedings of the Workshop on Software Engineering for Knowledge-Based Systems at the Twelfth International Joint Conference on Artificial Intelligence, Sydney, August, 1991.

(JoVe90) Jorysz, H., Vernadat, F.
"CIM OSA Part 1: Total Enterprise Modeling and Function View."
International Journal Computer Integrated Manufacturing,
Volume 3, Numbers 3 and 4, May, August, 1990.

(Ka-etal88) Kaiser, G., et al.
"Database Support for Knowledge-Based Engineering Environments."
IEEE Expert, Volume 3, Number 2, Summer, 1988.

(Ke90) Kerschberg, L.
"Expert Database Systems: Knowledge/Data Management Environments for Intelligent Information Systems."
Information Systems, Volume 15 Number 1, 1990.

(Ker90) Kerry, R.
Integrating Knowledge-based and Database Management Systems.
Ellis Horwood Limited, New York, 1990.

(Ki-etal88) Kim, W., et al.
"Integrating an Object-Oriented Programming System With a Database System."
OOPSLA 88 Proceedings, September, 1988.

(KiMa91) Kiernan, G., de Maindreville, C.
"A Rule Language Compiler for SQL Database Servers."
Data & Knowledge Engineering,
Volume 7, Number 2, December, 1991.

(KiMaSi90a) Kiernan, G., de Maindreville, C., Simon E.
"Making Deductive Database a Practical Technology: A Step Forward."
l'Institut National de Recherche en Informatique et en Automatique, 1990.

(KiMaSi90b) Kiernan, G., de Maindreville, C., Simon E.
"Making Deductive Database a Practical Technology: A Step Forward."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 19, Number 2, June, 1990.

(KnJa90) Knaus, R., Jay, C.
"Transporting Knowledge Bases: a Standard."
AI Expert, November, 1990.

(KoSi86) Korth, H., Silberschatz, A.
Database System Concepts.
McGraw-Hill Book Company, New York, 1989.

(KoSp88) Korth, H., Speegle, G.
"Formal Model of Correctness Without Serializability."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 17, Number 3, September, 1988.

(KrZa88) Krishnamurthy, R., Zaniolo, C.
"Optimization in a Logic Based Language for Knowledge and Data Intensive Applications."
Proceedings from the International Conference on Extending Database Technology, Venice, March, 1988.

(Ku91) Kumar, A.
"A New Approach For Conflict Resolution And Rule Processing In A Knowledge-based System."
Cornell University - in process, 1991.

(La-etal90) Lambrichts, E., et al.
"Integration of Functions in Logic Database Systems."
Data & Knowledge Engineering,
Volume 5, Number 3, September, 1990.

(Le89) Lenat, D.
"Ontological Versus Knowledge Engineering."
IEEE Transactions on Knowledge and Data Engineering,
Volume 1, Number 1, March, 1989.

(Le-etal90) Lenat, D., et al.
"CYC: Towards Programs With Common Sense."
Communications of the ACM,
Volume 33, Number 8, August, 1990.

(LeGu90) Lenat, D., Guha, R.
Building Large Knowledge-Based Systems.
Addison-Wesley Publishing Company, Inc.,
Reading, 1990.

(LeWu89) Leong, S., Wu, F.
"An AI Approach in Transporting Software."
IEEE Proceedings of SOUTHEASTCON '89,
Columbia, April, 1989.

(Li83) Linton, M.
"Queries and Views of Programs Using a Relational Database System."
PhD Thesis,
University of California, Berkley, 1983.

(LiTi88) Litwin W., Tirri, H.
"Flexible Concurrency Control Using Value Dates."
Institut National de Recherche en Informatique et en Automatique, Le Chesnay, 1988

(LoCuFa89) Lovegrove, G., Curtis, G., Farrar, R.
"Welding Advisory System for Process Selection 'WASPS'."
Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,
Tullahoma, June, 1989.

(Ma90) Mattos, N.
"An Approach to DBS-based Knowledge Management."
Proceedings of the First Workshop on Information Systems and Artificial Intelligence: Integration Aspects, Ulm, March, 1990.

(MaSh90) Mannino, M., Shapiro, L.
"Extensions to Query Languages for Graph Traversal Problems."
IEEE Transactions on Knowledge and Data Engineering, September, 1990.

(McDa89) McCarthy, D., Dayal, U.
"The Architecture of an Active Database Management System."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 2, June, 1989.

(McWe91) McLean, S., Weise, C. .
"Digress: A Deductive Interface to a Relational Database."
Journal of the American Society for Information Science, Volume 42, Number 1, January, 1991.

(Me91) Merrett, E.
"Relixpert - An Expert System Shell Written in a Database Programming Language."
Data & Knowledge Engineering,
Volume 6, Number 2, March, 1991.

(Mi87) Miranker, D.
"TREAT: A Better Match Algorithm for AI Production Systems."
Proceedings of the National AAAI Conference on Artificial Intelligence, 1987.

(Mi-etal88) Minker, J., et al.
Deductive Data Bases and Logic Programming.
Morgan Kaufmann Publishers,
Los Altos, 1988.

(Mo91) Morizet-Mahoudeaux, P.
"Maintaining Consistency of a Database During Monitoring of an Evolving Process by a Knowledge-Based System."
IEEE Transactions on Systems, Man and Cybernetics,
Volume 1, Number 1, January/February, 1991.

(Mu89) Murray, W.
"Control for Intelligent Tutoring Systems: A Blackboard-based Dynamic Instructional Planner,"
AI Communications,
Volume 2, Number 2, June, 1989.

(MyBr90) Mylopoulos, J., Brodie, M.
"Knowledge Bases and Databases: Current Trends and Future Directions."
Proceedings of the First Workshop on Information Systems and Artificial Intelligence: Integration Aspects, Ulm, March, 1990.

(Ne-etal91) Neches, R., et al.
"Enabling Technology for Knowledge Sharing."
AI Magazine, Volume 12, Number 3, 1991.

(NuLiKo90) Nurcan, S., Lei, L., Kouloumdjian, J.
"Integrating Database Technology and Logic Programming."
Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems - IEA/EIA 90, 1990.

(Pa86) Partridge, D.
Artificial Intelligence: Applications in the Future of Software Engineering.
Ellis Horwood Limited, Chichester, 1986.

(Pa-etal90) Park, S., et al.
"Integrating Inductive Learning and Simulation in Rule Based Scheduling."
Expert Systems in Engineering: Principles and Applications.
International Workshop Proceedings,
Vienna, September, 1990.

(PaPo91) Paul, V., Polamraju, R.
"Rule Management and Inferencing in Relational Databases."
IEEE Proceedings of SOUTHEASTCON '91,
Williamsburg, April, 1991.

(Py80) Pyster, A.
Compiler Design and Construction.
Van Nostrand Reinhold Company, New York, 1980.

(QaKi92) Qadah, G., Kim, J.
"The Processing of a Class of Transitive Closure Queries on Uniprocessor and Shared-Nothing Mulitprocessor Systems."
Data & Knowledge Engineering,
Volume 8, Number 1, April, 1992.

(RaTh90) Rahm, E., Thomasian, A.
"Distributed Optimistic Concurrency Control for High Performance Transaction Processing."
PARBASE90 International Conference on Databases, Parallel Architectures and Their Applications,
Miami, March, 1990.

(ReLe91) Reifman, J., Lee, J.
"Reactor Diagnostics Rule Generation Through Statistical Pattern Recognition."
Nuclear Science and Engineering,
Volume 107, April, 1991.

(Ri90) Riededsel, J.
"Knowledge Management: An Abstraction of Knowledge Base and Database Management Systems."
Proceedings of the Fifth Annual AI Systems in Government Conference,
Washington, D. C., May, 1990.

(Ro-etal86) Rosenthal, A., et al.
"Traversal Recursion: A Practical Approach to Supporting Recursive Applications."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 15, Number 5, June, 1986.

(RoSt87) Rowe, L., Stonebraker, M.
"The POSTGRES Data Model."
Proceedings of the Thirteenth International Conference on Very Large Databases.
Morgan Kaufmann Publishers, Inc., Los Altos, 1987.

(RoYe90) Rothi, J., Yen, D.
"Why American Express Gambled on an Expert Database."
Information Strategy: The Executive's Journal,
Spring, 1990.

(SaWi90) Salvini, S., Williams, M.
"Knowledge Management for Expert Systems."
IEEE Colloquium on Knowledge Engineering,
(Digest Number 77), May, 1990.

(Se-etal87) Sellis, T., et al.
"Expert Database Systems: Efficient Support for Engineering Environments."
Data and Knowledge Engineering,
Volume 3, Number 2, September, 1988.

(SeLi90) Sellis, T., Lin, C,
"Performance of DBMS Implementations of Production Systems."
Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, Herndon, 1990.

(SeLiRa88) Sellis, T., Lin, C, Raschid, L.
"Implementing Large Production Systems in a DBMS Environment: Concepts and Algorithms."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 17, Number 3, September, 1988.

(SeLiRa90) Sellis, T., Lin, C., Raschid, L.
"Coupling Production Systems and Database Systems: A Homogeneous Approach."
University of Maryland Computer Science Technical Report Series, 1990 revision.

(SeZh91) Segev, A., Zhao, J.
"Evaluation of Rule Processing Strategies In Expert Databases."
Proceedings of the Seventh International Conference on Data Engineering, Kobe, April, 1991.

(Sh87) Shaw, P.
"Closure Expressions."
ANSI X3H2-87-330, 1987.

(Sh88a) Shaw, P.
"Closure Expressions: Handling Cycles."
ANSI X3H2-88-63, 1988.

(Sh88b) Shaw, P.
"A Generalization of Recursive Expressions for Non-linear Recursion of Fixed Degree."
ANSI X3H2-88-93, 1988.

(Sh90) Shintani, T.
"Knowledge Table: An Approach to Speeding Up the Search for Relational Information in Knowledge Base."
Journal of Information Processing,
Volume 13, Number 4, 1990.

(Sk89) Skarra, A.
"Concurrency Control for Cooperating Transactions in an Object-oriented Database."
ACM SIGPLAN Notices. Volume 24, Number 4, April, 1989.

(So92) Sowa, J.
"Conceptual Graphs as a Universal Knowledge Representation."
Computers And Mathematics With Applications,
Volume 23, Numbers 2-5, January-March, 1992.

(St-etal88) Stonebraker, M., et al.
"The Design of XPRS."
Proceedings of the 14th VLDB Conference,
Los Angeles, August, 1988.

(StHaHo87) Stonebraker, M., Hanson, E., Hong, C.
"The Design of the Postgres Rules System."
The Proceedings of the Third International Conference on Data Engineering,
Los Angeles, February, 1987.

(StHaPo88) Stonebraker, M., Hanson, E., Potamianos, S.
"The POSTGRES Rule Manager."
IEEE Transactions on Software Engineering,
Volume 14, Number 7, July, 1988.

(StHe88) Stonebraker, M., Hearst, M.
"Future Trends in Expert Database Systems."
Proceedings From the Second International Conference on Expert Database Systems,
Tysons Corner, April, 1988.

(StHePo89) Stonebraker, M., Hearst, M., Potamianos, S.
"A Commentary on the Postgres Rules System."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 3, September, 1989.

(StSeHa87) Stonebraker, M., Sellis, T., Hanson, E.
"An Analysis of Rule Indexing Implementations In Database Systems."
Kerschberg, L. (ed).
Proceedings from the First International Conference on Expert Database Systems.
The Benjamin/Cummings Publishing Company, Inc., Menlo Park, 1987.

(StWoAn83) Stonebraker, M., Woodfill, J., Andersen, E.
"Implementation of Rules in Relational Database Systems."
Database Engineering,
Volume 6, Number 4, December, 1983.

(SuPa90) Su, S., Park, J.
"A Knowledge Representation Scheme and a Knowledge Derivation Mechanism for Achieving Rule Sharing Among Heterogeneous Expert Systems."
Proceedings of the International Conference on Database and Expert Systems Applications,
Vienna, August, 1990.

(Sw89) Swift, T.
"RDL: A Shell for Expert Systems That Deals With Large Data Bases."
Eighth International Workshop. Expert Systems and Their Applications, Volume 3, 1989.

(TaScYa91) Taglieri, M., Scally, M., Yablonsky, D.
"The Initiative for Managing Knowledge Assets (IMKA): A Review and Look Forward."
Eleventh International Conference. Expert Systems and Their Applications, Volume 1, 1991.

(TaSr92) Tan, J., Srivastava, J.
"Efficient Rule Matching in Large Scale Rule Based Systems."
Proceedings of the Twenty Fifth Hawaii International Conference on System Sciences,
Kauai, January, 1992.

(ToBl88) Tompa, F., Blakeley, J.
"Maintaining Materialized Views Without Accessing Base Data."
Information Systems, Volume 13, Number 4, 1988.

(Ts88) Tsur, S.
"LDL - A Technology for the Realization of Tightly Coupled Expert Database Systems."
IEEE Expert, Fall, 1988.

(Tw88) Twine, S.
"Representing Facts in KEE's Frame System."
Meersman, R., Shi, Z., Kung, C. (eds).
Artificial Intelligence in Information Systems (DS-3). Proceedings of the IFIP TC2/TC8/WG 8.1 Working Conference on the Role of Artificial Intelligence in Database and Information Systems.
North Holland, Amsterdam, 1988.

(Tz88) Tzvieli, A.
"On the Coupling of a Production System Shell and a DBMS."
Proceedings of the Third International Conference on Data and Knowledge Bases, 1988.

(Va87) Valduriez, P.
"Join Indices."
ACM Transactions on Database Systems,
Volume 12, Number 2, June, 1987.

(VaCoVa90) Van Brussel, H., Cottrez, F. Valckenaers, P.
"A Scheduling Expert System for Flexible Assembly."
Proceedings of the 1990 IEEE International Conference on Robotics and Automation,
Cincinnati, May, 1990.

(VaKh89) Valduriez, P., Khoshafian S.
"Transitive Closure of Transitively Closed Relations."
Kerschberg, L (ed).
Proceedings from the Second International Conference on Expert Database Systems.
The Benjamin/Cummings Publishing Company, Inc.,
Redwood City, 1989.

(Wa-etal87) Walker, A, et al.
Knowledge Systems and Prolog.
Addison-Wesley Publishing Company, Inc.,
Reading, 1987.

(WaYu90) Wang, Z., Yungui, C.
"Design and Evaluation of a Relational Knowledge Base Prototype Machine."
Future Generation Computer Systems,
Volume 6, Number 1, June, 1990.

(WhStLu90) Whitehall, B., Stepp, R., Lu, S.
"Knowledge-Based Learning: Integrating Acquisition and Learning."
Proceeding of the Third International Conference on Industrial Applications of Artificial Intelligence & Expert Systems, Charleston, July, 1990.

(WiCoLi91) Widom, J., Cochrane R., Lindsay, B.
"Implementing Set-Oriented Production Rules as an Extension to Starburst."
IBM Research Report, Februrary, 1991.

(WiFi89) Widom, J., Finkelstein, S.
"A Syntax and Semantics for Set-Oriented Production Rules in Relational Database Systems."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 18, Number 3, September, 1989.

(Wo-etal91) Wolfson, O., et al.
"Incremental Evaluation of Rules and Its Relationship to Parallelism."
ACM SIGMOD International Conference on Management of Data,
SIGMOD Record, Volume 20, Number 2, June, 1991.

(Xe90) Xenakis, J.
"Business Intelligence."
InformationWeek, August, 1990.

(YoIt86) Yokota, H., Itoh, H.
"A Model and an Architecture for a Relational Knowledge Base."
Proceedings of the 13th International Symposium on Computer Architecture, Tokyo, 1986.

(YoKiHa89) Yokota, H., Kitakami, H., Hattori, A.
"Term Indexing for Retrieval by Unification."
Proceedings of the 5th International Conference on Data Engineering, Los Angeles, 1989.

(Za90) Zaniolo, C.
"Architecture of Deductive Database Systems."
Digest of Papers: Thirty-fifth IEEE Computer Society International Conference,
San Francisco, February, 1990.

(ZhHi90) Zhang, Y., Hitchcock, P.
"Coupling Prolog to a Database Management System."
Information Systems, Volume 15, Number 6, 1990.

(ZhKa89) Zhong, X., Kambayashi,Y.
"Timestamp Ordering Concurrency Control Mechanisms for Transactions of Various Length."
Proceedings of Third International Conference, Foundations of Data Organization and Algorithms,
Paris, June, 1989.