

# Expert System Integrity Maintenance for the Retrieval of Data From Engineering Databases

William J. Rasdorf<sup>1</sup>, P.E. and TsoJen Wang<sup>2</sup>

## Abstract

This paper describes a mechanism that enables one to automatically monitor and evaluate engineering design data retrieval. It presents a framework, associated with a relational database management system, that combines a database with a set of design specification constraints that govern the retrieval of data from the database. This requires a database that is able to store all of the data normally associated with engineering design as well as a framework that is able to store all the domain-dependent constraints imposed upon the use of the design data. Such a framework has been successfully constructed and is described herein. An example, based on the AISC Specification, is presented and the performance of the framework is discussed.

## Introduction

The 1980's have been termed the information age. Data is collected in great quantities for use in either a centralized or a distributed manner. The demand for more data and for greater access to data is increasing. End user computing is coming of age and end users want data available to them from large centralized data sources (downloading), they want to incorporate their data into and influence the content of those sources (uploading), and they want access to and flexible transfer between those data sources (networking).

The correctness of data has been a concern of researchers for a number of years. Mechanisms and procedures have been developed to maintain the correctness of the stored content of a widely available database. However, there are no mechanisms in existence for insuring the integrity of the ways which data are retrieved. With the ever increasing availability of data, steps must be taken to develop new methods of guaranteeing the integrity of data retrieval operations. Clearly this is a very difficult problem that requires, for its solution, an understanding of the variety of ways that data are retrieved.

This paper advocates the need to develop a mechanism, referred to as a **Query Monitor** [29], that resides between a DBMS and its users and which serves as an experienced expert consultant to the users. The mechanism embodies the knowledge of the specific domain of interest (e.g. allowable stresses in steel members) so that whenever a user retrieves data

<sup>1</sup>Assistant Professor of Civil Engineering and Computer Science, North Carolina State University, Box 7908, Raleigh, NC 27695.

<sup>2</sup>Graduate Research Assistant, Computer Studies Program, North Carolina State University, Box 8207, Raleigh, NC 27695.

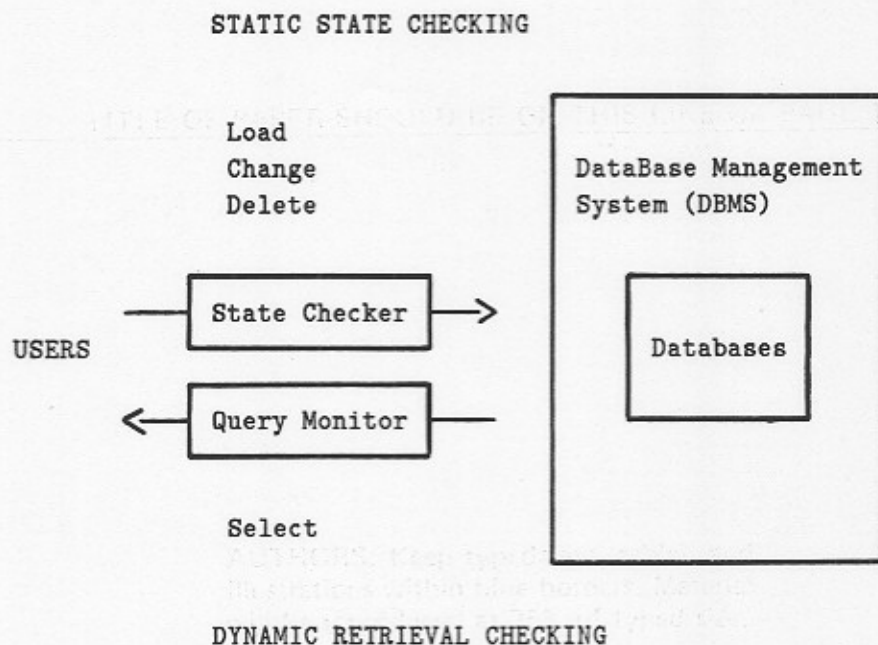


Figure 1: Database Integrity Checking Mechanisms

from the database the mechanism is activated, it checks the intended use of the data, derives relevant interim data items, and provides the correct data to satisfy the user request.

Previous work in the area of integrity checking concentrated on checking data item values as they were LOADED, CHANGED, or DELETED [7,10,17,21,24,30]. The earlier objective was to maintain the integrity of the stored data, i.e., to guarantee the correctness of any static state of the database (see Figure 1). This paper advocates the additional position that the data being retrieved from the database should be checked as well as the data being inserted into the database. It deals, therefore, with data being SELECTed for use.

In short, the mechanism presented here interprets data requests, applies the appropriate constraints to guide in the derivation and/or selection of the data, and provides the correct data. Its understanding of the semantics of the constraints it contains insures the validity of data retrieval and therefore makes a significant contribution to the engineering problem-solving environment.

## Relational Database Management Systems

An engineering application often involves a number of databases which are stored on secondary storage devices such as disks and tapes. A DataBase Management System (DBMS) is a collection of software which manages database storage and, more importantly, provides users and applications with access to databases [3,6,27].

A data model defines the overall logical structure of a database [28]. It provides a structural framework into which the data is placed. A relational data model is a single level

model consisting of a collection of interrelated relations represented in two-dimensional tabular form as shown in Figure 2 [22]. Associated with the relations is a set of operators that perform the insertion, deletion, modification, and retrieval of data.

Figure 2 illustrates the structure of a relation. The rows of a relation are called tuples and its columns are called attributes. The domain of an attributes is the set of allowable values the attribute may possess. The values in all of the attributes of Figure 2 are drawn from either the domain of characters, the domain of positive real numbers, or the domain of positive integer numbers. Each tuple represents an individual component and contains a value for each attribute. All tuples are distinct; duplicates are not permitted [9,18]. Tuples and attributes have no order; they may be arbitrarily interchanged without changing the data content and the meaning of the relation. Tuples are accessed by means of a key, a single attribute or a combination of attributes that uniquely identifies a tuple [6].

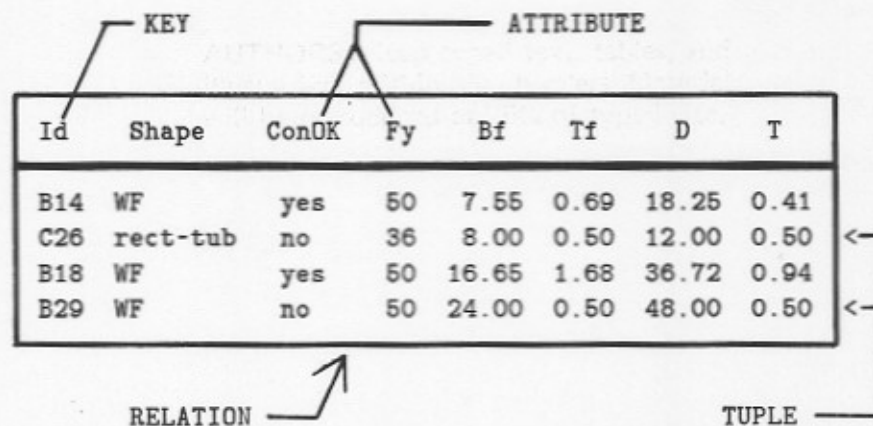


Figure 2: The Structure of Relations

A standard shorthand notation to represent relations is as follows:

RELATIONname (KEYATTRIBUTEname, ATTRIBUTE2name, ...)

with the MEMBER relation of Figure 2 being represented as:

MEMBER(Id, Shape, ConOK, Fy, Bf, Tf, D, T)

where the data items have the following meaning:

- Id* = An identification name for the structural member
- Shape* = Shape of the structural member
- ConOK* = An attribute to record whether or not the connection between the web and flange of the structural member is continuous

$F_y$	=	Steel yield stress (psi)
$B_f$	=	Width of the flange of the structural member (in)
$T_f$	=	Thickness of the flange of the structural member (in)

The name of the relation is listed first, followed in parentheses by the names of all of its attributes. The underlined attribute of a relation is the key.

## Expert Systems

Expert systems are computer programming systems that solve domain-dependent problems with the degree of expertise of a human specialist in the field [8]. Expert systems rely on the acquisition, representation, processing, verification, and use of knowledge [11]. Since knowledge is the central theme, expert systems are also called knowledge-based expert systems (KBES). The application of KBESs in engineering appears to be most promising [19,20,23].

An expert system consists of three main components: an inference engine, a knowledge base, and a representation of the problem at hand. The inference engine is a computer program that uses knowledge bases and a representation of the problem to draw conclusions. The knowledge base is a collection of facts, procedures, and rules gathered from an expert by a knowledge engineer. The expert is a person chosen for his or her high level of performance in the domain being captured, for example, structural analysis. The knowledge engineer specializes in extracting information from the expert through interviews, role-playing, and observation. The knowledge engineer then selects expert system tools and develops strategies to represent the decision processes of the expert in a computerized form.

Expert systems, as engineers know them, limit themselves to domain-specific knowledge rather than to general problem solving techniques. SACON [2], for example, dealt with structural analysis, using finite element techniques to simulate the behavior of a physical structure under various loading conditions. The more limited, or domain-specific, the problem, the greater its chances of being captured successfully in an expert system.

A production system is an expert system whose knowledge base consists of an unordered collection of basic units called production rules [4,5]. There are two parts, namely the condition and the action, to each production rule. In one scenario of operation the inference engine matches the conditions in each rule against instances of data from the data memory. It then selects a rule and executes the actions specified in that rule. Because of the nature of rule execution, production systems are very data-sensitive, i.e., the sequence of rule execution is determined by the presence or absence of data in data memory rather than by a predetermined rule ordering embedded in the system.

## Integrity Checking on Engineering Data Retrieval

Integrity checking on data retrieval can be addressed from two different perspectives. One is the issue of what the retrieved data is going to be used for. In general, the data stored in

a database must be used for the purpose for which it is intended. Information collected for one purpose is seldom of a form or nature to support other uses. For example, the AISC Specification for the Design, Fabrication, and Erection of Structural Steel for Buildings [1] is not structured to support unit pricing or project cost accounting. As a further example, design codes and standards, which can be cast in machine processible form and which perform design checking, cannot readily be used for design itself. In this paper we assume that the database user is knowledgeable in the domain and that the intended use of the data is appropriate for that domain.

The second issue of integrity checking on data retrieval occurs when there appear to be multiple answers to a query. In such a case the DBMS must be able to access the correct data item. This situation will occur when either one of the following conditions exists:

1. When providing the answer to the query involves the derivation of new data items based on a set of constraints and existing values in the database.
2. When the user has little or no knowledge of the domain and requires the aid of an intelligent retrieval system to direct him to the correct data.

While little work has been done with respect to the first condition, a significant amount of research has been conducted to develop user-friendly DBMS interfaces. Such interfaces utilize knowledge about the structure of their databases to provide retrieval suggestions and guidance to a user [15,26,31]. In particular, a great deal of effort has been devoted to developing data retrieval systems that utilize natural language processing techniques. These provide users with the ability to query databases using conversational English [12,13,16]; perform efficient text searching aimed at organizing the database in an efficient manner such that data can be accessed in the shortest time possible [13,14]; and, utilize query optimization which enables the DBMS to modify invalid or complex user queries to meet the specification of an underlying DBMS [15,26]. This paper proposes a method of dealing with the first condition listed above.

## Intelligent Engineering Data Retrieval System

Figure 3 illustrates the architecture of a production system integrity management system called Query Monitor designed to couple multiple domain-specific expert systems with relational design databases to ensure that the correct data is provided for design data retrieval operations. The Query Monitor consists of five major components: a Knowledge Base Selector, a System Knowledge Base, a Task Knowledge Base, an Inference Engine, and a Cache.

When the Query Monitor is activated, its Inference Engine (IE) constantly extracts knowledge from the System Knowledge Base (SKB) enabling it to operate the entire system. The Task Knowledge Base (TKB) is a repository of the domain-specific knowledge used by the IE to actually monitor user queries. At system startup the TKB is empty. Upon receipt of a user query the IE loads the TKB with the correct domain knowledge base from a set of external knowledge bases that were previously created by knowledge engineers and stored on secondary storage devices in ASCII file format. The loading of the TKB is achieved by sending to the Knowledge Base Selector (KBS) a load vector containing the name of the

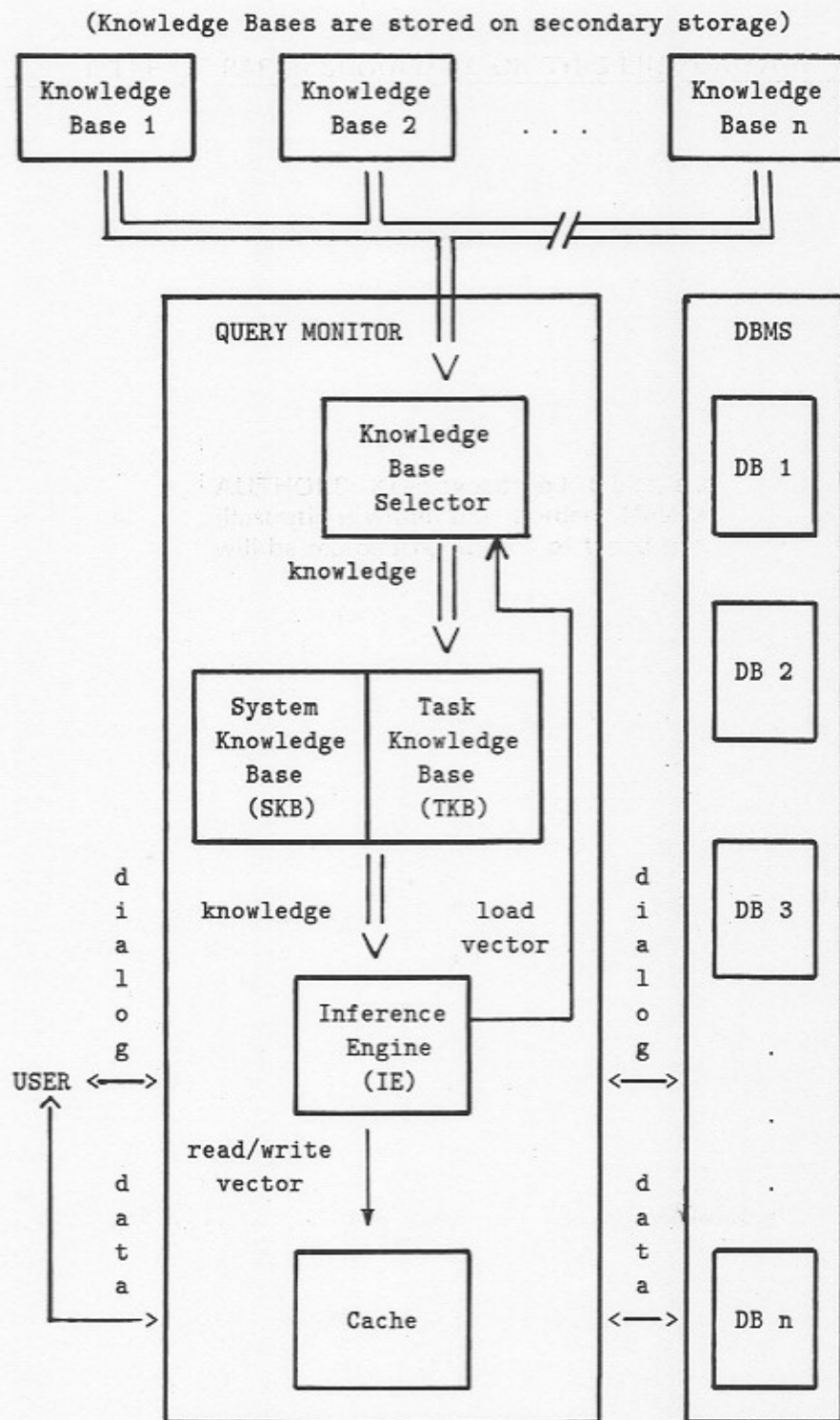


Figure 3: Architecture of the Query Monitor

knowledge base to be loaded. The KBS in turn accesses the corresponding knowledge base and writes it into the TKB. If there is a knowledge base in the TKB while a new one is being loaded, the old knowledge base is overwritten.

Upon completion of loading the TKB, the IE transfers control from the SKB to the TKB which communicates with the user and the DBMS, interpreting data requests, applying constraints, and retrieving the correct data from the databases. During the constraint evaluation process the Query Monitor stores the values of any intermediate data items that may be generated in a block of main memory referred to as the Cache thus enabling them to be reused efficiently. In this manner the content of the database is increased as a function of applying evaluation operations to previous data.

When the correct data has been provided to the user and he is satisfied, the IE switches control back to the SKB and waits for the next user query. If the interpretation of a subsequent query requires loading another knowledge base, the IE writes the data currently stored in the Cache into the appropriate database. The read/write vector, which is sent to the Cache when a read/write operation is needed, may contain one of two sets of data items: for communicating with the user, it contains the destination (user or DBMS), the type of operation (read or write), and the name and value of the data item(s); for communicating with the DBMS, it additionally contains the database name, the relation name, and the attribute name.

The arrow labeled "dialog" in Figure 3 is used to represent user queries, system prompts, user answers (non-data), procedure calls to the DBMS, and messages from the DBMS to the Query Monitor. The arrow labeled "data" represents data flow between users and the Query Monitor, and data flow between the Query Monitor and the DBMS.

The Query Monitor architecture is universal in the sense that it can be used in any discipline whose domain knowledge can be cast in rule form. An example will be presented in the next section to show how an appropriate set of production system rules can be generated in a systematic way.

One major advantage of the Query Monitor architecture is its flexibility. Because there is no limit, theoretically, on the number of knowledge bases, new knowledge bases can be added as the system grows. Additionally, existing knowledge bases can be modified or expanded and out-of-date knowledge bases can be removed. Another advantage of this architecture is that there is no need to modify the associated DBMS. The Query Monitor can be coupled with any existing DBMS by making minimal changes. Therefore it is also transportable.

## A Prototype Expert System (Query Monitor)

### Scope

During the process of designing a structural steel member the engineer proceeds by repeating the following simplified steps until a satisfactory design solution is achieved:

1. select an initial steel member (shape, composition, and size) based on initial design constraints.

2. compute the actual stresses that occur within the member using structural analysis methods.
3. compare the actual stresses from step 2 with the maximum allowable stresses given in the Specification.
4. if the allowable stresses are greater than the actual stresses, then the design is valid (although it may not be optimal); otherwise a larger member is chosen and steps 2 through 4 are repeated.

To perform step 3, the engineer will consult the AISC Specification to locate the provisions related to the design situation under consideration. He will then use the equations embodied in the provisions to compute the associated allowable stresses.

The AISC Specification addresses a number of different types of stresses which can occur within a structural steel member including tension, shear, compression, bending, and bearing. Depending upon constraints on shape, cross-section, loading, etc., any one of a number of equations can be used to determine the allowable stress for a specific structural steel member. Figure 4, for example, lists eight equations applicable to calculating allowable bending stress. The database problem arises when the engineer issues an  $F_b$  data retrieval request in order to conclude design step 3. The response to such a query requires a determination of which  $F_b$  is applicable and involves the subsequent derivation of its value.

A computer program has been developed to implement the Query Monitor framework described in the previous section and to investigate data retrieval integrity issues such as those encountered in step 3 of the structural steel member design scenario. By using the Query Monitor program engineers can acquire the correct formula and value of the allowable bending stress for structural steel members of WF, H, and rectangular-tubular shapes. The data items used in the Query Monitor are listed in Table 1.

### Rule Generation

Design specifications and model building codes are guidelines which govern design in a specific application area. The AISC Specification, for example, governs the design, fabrication, and erection of structural steel for buildings [1].

Although design specifications are written by groups of domain experts, they rarely exist in a form that can be directly used in a knowledge base. The rules and guidelines governing design are embedded within the text of a specification and must be extracted and recast in machine processible form. Figure 5 illustrates an approach to deriving production rules from specifications and codes that utilizes decision tables to facilitate the extraction of data items and the constraint relationships between them (conditions). Figure 6 shows a sample decision table corresponding to section 1.5.1.4 of the AISC Specification [1]. The upper-left portion of the table contains a set of conditions. The upper-right portion indicates whether (Y) or not (N) a condition is satisfied. The lower-left portion contains a set of actions to be performed. The lower-right portion indicates which action is to be taken if all of the corresponding conditions in a given column are matched. The rules relating the conditions and actions of a decision table have a direct correspondence to production system rules.

Resultant Data Items	Symbol
the formula (bending) of the H member	Fb
the formula (bending) of the WF member	Fb
the formula (bending) of the rect-tub member	Fb
Intermediate Data Items	Symbol
the area of the compression flange	Af
the bending coefficient	Cb
the compression flange is ok	FlangeOK
the compression formula for exceptions	Except_Comp
the depth thickness ratio	D_T_Ratio
the depth thickness ratio is ok	D_T_RatioOK
the first compression bending for exceptions	First_Bending
the larger end moment	M2
the laterally unsupported length is ok	LcOK
the second compression bending for exceptions	Second_Bending
the smaller end moment	M1
the tension formula for exceptions	Except_Ten
the width thickness ratio	Bf_Tf_Ratio
the width thickness ratio for exceptions is ok	Bf_Tf_RatioOK
Basic Data Items	Symbol
the axis the member is being bent about	Axis
the computed axial stress	Fa
the connection of web and flange is ok	ConOK
the depth of the member	D
the largest moment within an unbraced length of the member	Mu
the laterally unsupported length of the compression flange	Lc
the moment at the left end of an unbraced length of the member	Ml
the moment at the right end of an unbraced length of the member	Mr
the radius of gyration	Rt
the shape of the member	Shape
the steel yield stress	Fy
the thickness of the compression flange	Tf
the thickness of the web	T
the width of the compression flange	Bf

Table 1: Query Monitor Data Items for AISC Specification Section 1.5.1.4

$$F_b = 0.60F_y$$

$$F_b = 0.66F_y$$

$$F_b = 0.75F_y$$

$$F_b = \frac{12 \times 10^3 C_b}{ld/A_f}$$

$$F_b = \frac{170 \times 10^3 C_b}{(l/r_T)^2}$$

$$F_b = F_y \left[ 1.075 - 0.005 \left( \frac{b_f}{2t_f} \right) \sqrt{F_y} \right]$$

$$F_b = F_y \left[ 0.79 - 0.002 \left( \frac{b_f}{2t_f} \right) \sqrt{F_y} \right]$$

$$F_b = \left[ \frac{2}{3} - \frac{F_y(l/r_T)^2}{1530 \times 10^3 C_b} \right] F_y$$

Figure 4: Formulas to Determine Allowable Bending Stress

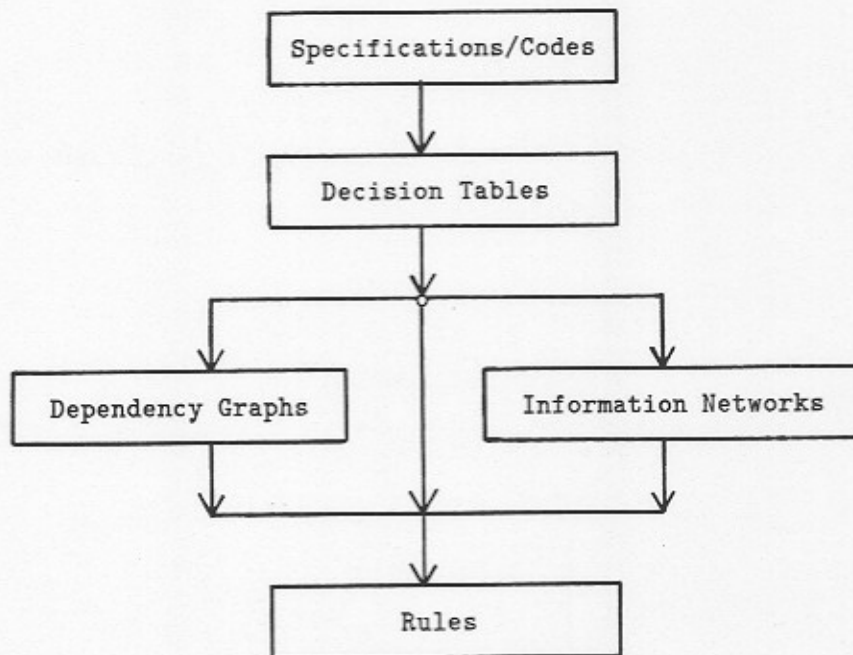


Figure 5: Generating Production Rules from Design Specifications

	(1)	(2)	(3)	(4)	(5)	(6)
Axis = major	Y	Y	Y	Y	Y	N
ConOK	Y	N	Y	Y	Y	Y
Bf_Tf_RatioOK	Y	Y	N	Y	Y	Y
D_T_RatioOK	Y	Y	Y	N	Y	Y
LcOK	Y	Y	Y	Y	N	I
Fb = 0.66 * Fy	Y					Y
Except_Ten		Y	Y	Y	Y	
Except_Comp		Y	Y	Y	Y	

Figure 6: Decision Table for AISC Specification Section 1.5.1.4

The first column in the sample decision table, for example, can be recast in production rule format as follows:

**If** the axis about which a member is being bent = major **and**  
the connection of the web and flange is continuous **and**  
the width thickness ratio for exceptions is ok **and**  
the depth thickness ratio is ok **and**  
the laterally unsupported length is ok  
**Then** the allowable bending stress =  $0.66F_y$

Two additional production rules can be extracted from the decision table to complete its production system representation:

**If** the axis about which a member is being bent = minor **and**  
the connection of the web and flange is continuous **and**  
the width thickness ratio for exceptions is ok **and**  
the depth thickness ratio is ok  
**Then** the allowable bending stress =  $0.66F_y$

**If** the axis about which a member is being bent = major **and**  
(the connection of the web and flange is not continuous **or**  
the width thickness ratio for exceptions is not ok **or**  
the depth thickness ratio is not ok **or**  
the laterally unsupported length is not ok)  
**Then** the allowable bending stress =  
the compression formula for exceptions **and**  
the tension formula for exceptions

## Implementation

M.1 was the production system building tool used to develop the Query Monitor. The major components of M.1 include an inference engine, a knowledge base, and a cache [25]. The M.1 inference engine adopts a goal-driven, backward-chaining search strategy. The primary objective of the M.1 inference engine execution cycle is to satisfy a goal by matching the premises of its rules against the data in the cache. If a goal cannot be satisfied immediately, the IE will try to satisfy its subgoals. The splitting of goals continues until the subgoals are satisfied. The IE will then fuse the results of the subgoals to achieve the initial goal.

Because of its structure M.1 is particularly suited to solving structured selection problems, i.e., problems where the solution is chosen from a finite predefined set of solutions. The selection of the correct formula to determine values for  $F_y$  is such a problem. The implementation of the Query Monitor program is by no means restricted to one language or tool, but a language or tool selection decision should be based on the characteristics of the particular application [4,5,11]. The merits of using M.1 or any other expert system framework are not considered here.

Two limitations currently restrict the Query Monitor. To develop it beyond the prototype stage, the following program extensions must be achieved:

1. Storing the basic data items (the dimensions of the member, the yield stress, the laterally unbraced length, etc.) in a database.
2. Establishing the communication link between the Query Monitor and the DBMS.

In the present implementation these extensions were replaced by program interaction with the user, i.e., the user provided basic data item values. The intermediate and the resultant data items were then derived by the system. They were not rewritten back into an underlying database. The System Knowledge Base and the multiple external knowledge bases were built by splitting the M.1 rule base into two mutual exclusive smaller rule bases and explicitly switching between them during the consultation process. Sample runs of the system have shown that the Query Monitor framework is both practical and feasible. A complete listing of the knowledge base and several sample interactive sessions can be found in the *Query Watcher Users Guide* [29].

## Summary

An obvious trend in computer use in engineering design is towards a higher level of integration of computer applications. The data exchanged and shared among applications and designers is the key to integration. The approach advocated in this paper for achieving integration is to continue to build on the highly developed areas of DBMS's and ES's, adding those specific extensions necessary for engineering design.

Integrity of both the stored data and its retrieval is critical. The maintenance of integrity for engineering data needs must be achieved on both of these fronts. The first requires the implementation of constraints as functions or procedures that guarantee the correctness of

the static state of the database. The second requires a constraint semantics interpreting mechanism to ensure the correctness of data retrieval requests, i.e., the correctness of the information that passes to the user. This second view is essential to guaranteeing proper engineering data use.

A knowledge-based consultant for data retrieval provides accurate constraint interpretation and serves as an active agent in correctly guiding the data retrieval process. The need for such an expert-like consulting mechanism to deal with design constraint semantics will become more acute as the trend towards more widely spread use of engineering databases continues.

## Acknowledgement

Portions of this work were sponsored by the National Aeronautics and Space Administration under Grant NAG-1-604 entitled "Constraint Semantics in Engineering Database Use" and by the National Science Foundation under Grant CEE-8319869 entitled "Generative Database Systems for Structural Engineering Design." The support of these organizations is gratefully acknowledged.

## References

1. American Institute of Steel Construction, Inc. *Manual of Steel Construction*, Eighth Edition, American Institute of Steel Construction, Inc., Chicago, IL, August (1978).
2. J. Bennett, L. Creary, R. Englemore, and R. Melosh. "SACON: A Knowledge-Based Consultant for Structural Analysis," Stanford Heuristic Programming Project Report HPP-78-23, Stanford University, Stanford, CA, September (1978).
3. C. L. Blackburn, O. O. Storaasli, and R. E. Fulton. "The Role and Application of Database Management in Integrated Computer-Aided Design," *Journal of Aircraft*, Volume 20, Number 8, Pages 717-725, American Institute of Aeronautics and Astronautics, August (1983).
4. L. Brownston, R. Farrell, E. Kant and N. Martin. *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, Reading, MA (1985).
5. B. Buchanan and E. H. Shortliffe, (eds.). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA (1984).
6. C. J. Date. *An Introduction to Database Systems*, Volume 1, Fourth Edition, Addison Wesley, Reading, MA (1986), and Volume 2, First Edition, Addison Wesley, Reading, MA (1983).
7. C. M. Eastman and S. J. Fenves. "Design Representation and Consistency Maintenance Needs in Engineering Database," *Engineering and Scientific Data Management*, Pages 1-18, NASA Conference Publication 2055, Langley, VA (1979).

8. E. A. Feigenbaum. "Knowledge Engineering: The Applied Side of Artificial Intelligence," Stanford Heuristic Programming Project Report HPP-80-21, Stanford University, Stanford, CA (1980).
9. S. J. Fenves and W. J. Rasdorf. "Role of Database Management Systems in Structural Design," *Proceedings of the IABSE Colloquium on Informatics in Structural Engineering*, Pages 229-242, International Association of Bridge and Structural Engineers, Bergamo, Italy, October (1982).
10. S. J. Fenves and W. J. Rasdorf. "Treatment of Engineering Design Constraints in a Relational Database," *Engineering With Computers*, Volume 1, Number 1, Pages 27-37, Springer-Verlag, Spring (1985).
11. F. Hayes-Roth, D. A. Waterman and D. B. Lenat, (eds.). *Building Expert Systems*, Addison-Wesley, Reading, MA (1983).  
Retrieval System," *Proceedings of the Third International Conference on Artificial Intelligence and Information-Control Systems of Robots*, Pages 171-174, Smolenice, Czechoslovakia, June (1984).
13. M. Lebowitz. "Intelligent Information Systems," *Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pages 25-30, Association for Computing Machinery, Bethesda, MD, June (1983).
14. C. T. Meadow, T. T. Hewett and E. S. Aversa. "A Computer Intermediary for Interactive Database Searching - I. Design," *Journal of American Society of Information Science (USA)*, Volume 33, Number 5, Pages 325-332, September (1982).
15. P. F. Patel-Schneider, R. J. Brachman and H. J. Levesque. "ARGON: Knowledge Representation Meets Information Retrieval," *Proceedings of the First Conference on Artificial Intelligence Applications*, Pages 280-286, Institute of Electrical and Electronic Engineers and American Association of Artificial Intelligence, Denver, CO, December (1984).
16. M. J. Pazzani and C. Engelman. "Knowledge Based Question Answering," *Proceedings of the Conference on Applied Natural Language Processing*, Pages 73-80, Santa Monica, CA, February (1983).
17. W. J. Rasdorf. *Structure and Integrity of a Structural Engineering Design Database*, Technical Report DRC-02-14-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, April (1982).
18. W. J. Rasdorf and S. J. Fenves. "Organization of a Structural Engineering Design Database," *Proceedings of the Eighth Conference on Electronic Computation*, Pages 559-571, American Society of Civil Engineers, Houston, TX, February (1983).
19. W. J. Rasdorf and G. C. Salley. "Generative Engineering Databases - Toward Expert Systems," *Computers and Structures*, Volume 20, Number 1-3, Pages 11-15, Pergamon Press (1985).

20. W. J. Rasdorf and L. M. Parks. "Expert Systems and Engineering Design Knowledge," *Proceedings of the Ninth Conference on Electronic Computation*, Pages 28-42, American Society of Civil Engineers, Birmingham, AL, February (1986).
21. W. J. Rasdorf and T. E. Wang. "CDIS: An Engineering Constraint Definition and Integrity Enforcement System for Relational Databases," *Proceedings of the 1986 International Computers in Engineering Conference*, American Society of Mechanical Engineers, Chicago, IL, July (1986).
22. G. Sandberg. "A Primer on Relational Database Concepts," *IBM Systems Journal*, Volume 20, Number 1, Pages 23-40 (1981).
23. D. Sriram, et. al. "Applications of Expert Systems in Structural Engineering," *Proceedings of the Conference on Artificial Intelligence*, Oakland University, Rochester, MI, April (1983).
24. M. Stonebraker, J. Woodfill, and E. Anderson. "Implementation of Rules in Relational Database Systems," *Database Engineering*, Volume 6, Number 4, Pages 65-74, December (1983).
25. Teknowledge, Inc.. *M.1 Training Materials* (1984), *M.1 Reference Manual*, version 1.3 (1985), *M.1 Sample Knowledge Systems*, version 1.3 (1985), Teknowledge, Inc., Palo Alto, CA.
26. R. M. Tong, V. N. Askman and J. F. Cunningham. "RUBRIC: An Artificial Intelligence Approach to Information Retrieval," *Proceedings of the First International Workshop on Expert Database Systems*, Pages 360-377, Kiawah Island, SC, October (1984).
27. D. C. Tsichritzis and A. Klug, (eds.). "The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems," *Information Systems*, Volume 3, Number 3, Pages 173-191 (1978).
28. J. D. Ullman. *Principles of Database Systems*, Computer Science Press Inc., Potomac, MD (1980).
29. T. E. Wang and W. J. Rasdorf. *Query Watcher Users Guide*, Department of Civil Engineering and Program of Computer Studies, North Carolina State University, Raleigh, NC, October (1985).
30. G. A. Wilson. "A Conceptual Model for Semantic Integrity Checking," *Proceedings of the Sixth International Conference on Very Large Databases*, Montreal, Canada, October (1980).
31. G. P. Zarri. "Reseda, An Information Retrieval System Using Artificial Intelligence and Knowledge Representation Techniques," *Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pages 189-195, Association for Computing Machinery, Bethesda, MD, June (1983).