The Desirability of Embedding An Expert System Shell Within A Relational DBMS

by
John P. Coyne
Department of Management Science
George Washington University
Washington, D.C. 20052

Abstract
This article examines the desirability of embedding a rule based expert system shell entirely within a relational DBMS. It is argued that embedding a shell may have significant advantages beyond improved access to database tables.

1 Introduction
There is little doubt that expert systems shells continue to gain increasing acceptance as useful tools in the production of expert systems. One of the major advantages of such shells is that they enable the expert system developer to concentrate more on knowledge base development rather than on mundane hardware/software issues [1]. The majority of these shell products are rule based and provide for forward, backward and mixed chaining modes. Since so many expert systems require access to external data, virtually every shell provides some type of software interface to commercial DBMS packages. The robustness, flexibility and ease of use for such interfaces often becomes critical factors in shell selection.

This article examines the desirability of embedding a rule based expert system shell entirely within a relational DBMS. It is argued that such an approach has some significant advantages beyond the most obvious one of improved access to relational database tables.

2 Embedding a Shell

Exactly what is meant by "embedding a shell" within a relational DBMS? Essentially, all data associated with the shell is stored within relational tables. There are no external files of any kind associated with the shell. All information/data that was formerly managed by the shell's code is now contained in tables in the DBMS where it is accessed/updated by the shell's code via calls (typically using SQL commands) to the DBMS. Information such as rule names, rule components, author names, rule bases, rule priorities, certainty factors, attribute properties, values, justifications, goals, explanations, security, results, chaining priorities etc. are all contained in DBMS tables.

Exactly how many tables would such an arrangement require? There are many design alternatives that would depend on a number of factors. Certainly, such a table design is not a trivial problem although it should not be an inordinately complex problem depending on the functional capabilities of the shell that was embedded. For example, consider taking all of the source rules for all expert systems managed by the shell and placing them in a single table. In such an organization, the primary key for the table might be a concatenation of the expert system name and rule name. Nonkey attributes might include rule number, author, the actual rule itself, its priority, certainty, computational cost, author, and its explanation/basis. Additional tables could be used to contain attribute information, results etc. Finally some tables may be required that hold no permanent data but are used simply for intermediate results, pointers etc. during the inference process.

Given all of the shell's data in table form, the code for executing forward chaining and/or backward chaining of the rules would be external to the DBMS and would access the knowledge base...
through calls to the DBMS. The inference engine, user interface and other code would be written in a good system development language like C.

3 Advantages

Besides being interesting as a software design problem, are there any tangible and/or intangible benefits to embedding a shell in the manner just described? Consider the following:

3.1 Organizational Acceptance of Expert Systems Technology

Virtually every organization has adopted some form of relational database strategy. The relational data model is clearly the dominant model in commercial database systems and it appears unlikely that this situation will change soon despite interest in such topics as object oriented databases [2]. Perhaps the most straightforward way to introduce expert systems into an organization is to have expert system technology appears as a logical enhancement or logical extension to the already installed relational DBMS. In this way expert systems is not seen as a competing technology to relational DBMS. Additionally, many of the technical questions that delay the adoption of expert systems technology are avoided since the expert system is seen as a part of the database system. If one considers that a significant percentage of data processing professionals (who are in senior positions) completed their formal data processing education before expert systems technology became part of the academic curriculum, then presenting an expert system shell as an extension to the DBMS, rather than as a competing and somewhat mysterious technology, should reduce "technical anxiety" and truly minimize potential resistance due to unfamiliarity.

3.2 Access to DBMS tables

One of the first real world lessons that a knowledge engineer learns is that virtually every non-trivial expert system requires access to external database information. All commercial shells indicate that they provide convenient and easy-to-use interfaces to external DBMS packages. Unfortunately, the reality of the situation is that it is often a significant software engineering job to make the interface work. It is obvious that embedding the shell directly within a relational DBMS should make access to that DBMS a more straightforward task.

3.3 Portability

One of the practical reasons why relational database systems have become so popular is that so many of them are portable across various hardware/software platforms. While this has nothing to do with the relational model per se, it does reflect the industry's preoccupation with portability. By embedding a shell within a DBMS, expert systems built with the shell are as portable as the DBMS. Since DBMS selection usually involves a great deal of care concerning current and future hardware and software compatibility for an organization, it follows that an embedded shell will automatically be as well suited for the organization as the DBMS product chosen.

3.4 Security

For a knowledge engineering perspective, careful control of the updating of the rules for an expert system requires some level of security processing. This is particularly true when there are multiple domain experts. An embedded shell may make use of the SQL view mechanism in relational database systems. The view mechanism should be more than adequate for controlling rule updating and, if it is desired, controlling user access to the expert system.

3.5 Recovery

The updating of a knowledge base for an expert system is certainly less frequent than the updating requirements of a volatile database application. Nevertheless, the knowledge base is as susceptible to problems from system crashes as any other piece of software. An embedded shell can exploit all of the recovery procedures of a relational DBMS (typically a restore and roll forward procedure).

3.6 Multimedia Access

In order to enhance the user-system interaction component of an expert system, the value of multimedia is becoming recognized. What is more interesting is that a number of well known DBMS products are aggressively pursuing the incorporation of multimedia within their packages.
Products such as Informix, Oracle, Ingres and others all have some type of BLOB data type to handle text, images and other multimedia information.

If the shell is embedded in the DBMS, the shell can access these multimedia objects as easily as it accesses the formatted data in the database. Given access to such multimedia objects, a user-expert system dialogue could be significantly enhanced through the use of text, image, video and audio.

3.7 Integrity Processing for the DBMS

All of the above advantages have focused on what a relational DBMS can do for an expert system. However, the embedded shell may provide a commensurate, if not greater, benefits to the DBMS. The shell, if tightly integrated to the DBMS code, may provide a convenient mechanism for DBMS users to express integrity constraints as rules in an expert system. After a change is made to a database but before a COMMIT is issued, an expert system could forward chain to determine if any anomalies were present. If so, a rollback could be issued. To date, no DBMS vendor has developed a comprehensive integrity subsystem. However, if a rule based shell were available and tightly coupled to the DBMS, then perhaps the task of maintaining a specific database's integrity could be offloaded to a user written expert system.

3.8 Expert Systems Assistance for DBMS Users

The presence of an embedded shell may make it possible for the dbms vendor to deliver customized knowledge to the dbms users. Expert systems to assist in table design, normalization, SQL query formulation and other database activities could be made available to users of the DBMS. Users should not need to shuffle between separate packages when the expertise is "embedded" in the DBMS that they are using.

4 Disadvantages

Despite the advantages, some major obstacles remain to embedded shells becoming a common commercial reality. First, there is a well established installed base of shells that are external to any DBMS. From a business perspective these products must be maintained, enhanced and new releases offered on a periodic basis. Transitioning a widely sold stand-alone shell product to an embedded DBMS product would be an extremely difficult task from both a marketing and technical perspective. Second, many (if not all) shell vendors strongly insist that they already provide easy access to DBMS packages and any problems/deficiencies will be "fixed in the next release." Third, embedding a shell in a DBMS may provide the shell with easy access to DBMS X but may also make it more difficult for the shell to access DBMS Y if that became necessary. Fourth, from a maintenance perspective, shell vendors won't want to produce and maintain as many versions of their shells as there are popular relational database systems.

5 Conclusions

There are some distinct advantages to embedding a shell within a DBMS. However, shell developers may see it as less than optimal goal for the reasons cited above. For these reasons, it is unlikely that they will quickly embrace the concept. What is more likely is that a specific DBMS vendor may take the initiative and offer an embedded shell as an option to increase product differentiation and provide the basis for a flexible integrity subsystem. Additionally, the dbms vendor may then deliver (at perhaps an additional cost) specific expert systems to assist in relational table design, normalization, SQL query formulation, distributed database design and other related topics. Seen as an option to the dbms vendors product, the advantages of expert systems technology may be more reality incorporated into mainstream business applications.

References