

APPLYING SQL TECHNIQUES ON INFERENCE MECHANISMS OF EXPERT SYSTEMS

George DOBRE, Cosmin GRIGORESCU, Dorin
CARSTOIU, Simona GRIGORESCU

*Department of Machine Elements and Tribology and
Department of Computers and Industrial Informatics,
University "POLITEHNICA" of Bucharest,
Spl. Independentei 313, 77206 Bucharest, Romania
e-mail: geo@meca.omtr.pub.ro, cosming@ciid.pub.ro,
cid@aia.pub.ro, simona@ciid.pub.ro*

Abstract

The paper approach the implementation of the Structured Query Language (SQL) database technology in the strategies for forward or backward reasoning in the rule based Expert Systems (ES). The SQL technology, which is well known from the Database Environments, was recently applied in the new powerful object oriented software tools. Thus, the SQL technology becomes very attractive for the developers of the ES. After a short description of the main inference mechanism strategies linked with the SQL approach, a solution for implementing a SQL inference mechanism is given in the paper. This solution is discussed for each type of forward and backward chaining. A case study for an implemented forward chaining mechanism as a part of an ES elaborated by the authors is discussed. Finally, certain conclusions are presented.

1. Introduction

Nowadays, the tendency of using computer based decisional act is increasing. Artificial Intelligence (AI) is aimed to solve such problems in order to increase work efficiency in more or less intensive computational tasks. The ideal solution seems to be the capacity of the software to combine traditional pure computational tasks with decisional “human-like” tasks. In AI domain, a lot of strategies for combining these tasks were imagined, each strategy having it’s own knowledge representation format, deriving knowledge mechanism, tracing and explaining presentation forms.

In the class of AI software systems, Expert Systems (ES) are the ideal choice when dealing with both *formal knowledge* (data representation in numeric format) and *non-formal knowledge* (data representation in facts, rules, degrees of certainty, etc.) (Coyne[1]). These systems have the advantage of a modular structure and the capacity of easy adapting all “human-like” abstract data representation formats to ES suitable data representation formats.

In the structure of ES, the inference mechanism has an essential component, responsible for reasoning process. The design of the inference mechanism can be carried out using several strategies. Two strategies are well known and used, namely forward and backward chaining strategies.

The paper shows a solution for implementing both forward and backward reasoning strategies for rule based ES, using the Structured Query Language (SQL) database technology. This SQL technology is well known for Database Environments and was recently integrated in new powerful object oriented software tools. Therefore, in our opinion this technology comes of interest for ES shells, becoming very attractive for ES developers.

After a short description of the main inference mechanism strategies linked with the SQL approach, a solution for implementing a SQL inference mechanism is given, which is discussed for each type of chaining: forward and backward. A case study for an implemented forward chaining mechanism as a part of an ES is outlined. Finally, certain conclusions are withdrawn.

2. Inference Mechanisms Strategies

The structure of an ES contains an Inference Mechanism Module that implements the reasoning strategy, i.e. how the system derives

knowledge. Beside this module, a Knowledge Database Module that stores knowledge in a suitable format is needed (Carstoiu[2]). Several other modules need also be present in order to have a fully functional ES, e.g. Knowledge Acquisition Module, User Interface Module, Working Memory (Context), Trace and Explanation Module etc. A complete structure of an ES that works like a shell containing the inference mechanism is given below (fig. 1).

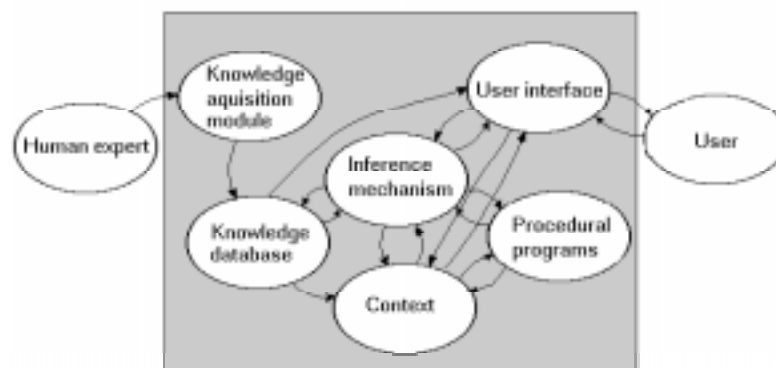


Fig. 1. *ES general structure [3]*

As it can be noticed, several interaction paths among the ES modules are needed in this ES environment. As a result of this multiple interaction, the inference mechanism has a central role for this environment.

Also, the knowledge representation influences how the inference mechanism derives the new information from the existent one. The most popular approach for knowledge representation is the rules based approach. Using this representation, propositions are stored as ES facts. At their turn, the facts are atoms of rules concatenated by logic connectives (as AND, OR).

For the rule based knowledge representation, there are two main strategies of deriving knowledge (inference strategies) (Durkin[3]):

1. forward chaining strategy;
2. backward chaining strategy.

Each one of these strategies has its own strengths and weaknesses. Below, several peculiarities of these strategies are enounced for getting a good image of the SQL technology application for each of these strategies.

The forward chaining strategy starts with some initial facts (assumed to be known) and tries deriving new facts from these ones by applying rules. At a moment, several rules may be triggered and the system must decide which ones to fire. In most applications, only one rule will be fired and a new cycle of the rules begins. The process of locating rules that can fire, choosing one and firing it is known as “recognise-resolve-act” cycle. The main problem (regarded as a disadvantage of this strategy) is that some rules ready to be fired may lead to opposite conclusions. The simplest solution of this problem is to give each rule a priority and fire the highest priority rule.

The backward chaining strategy is used to prove some goal or hypothesis. The goal rules (that contain goal facts in the THEN part) are examined and the IF part facts become subgoals. The process ends when a subgoal can be proved (the user points out that the subgoal is true or false). The main disadvantages of this strategy regard the situation when the user does not know the goal to be reached, so this strategy is not applicable, and, on the other hand, the number of rules involved in the recursion process is very large, making the inference very slow.

Forward chaining inference is preferred over backward chaining if either the goals are not known apriori or their number is very large. Designing a backward chaining system starts with defining system goals, while defining initial data is the starting point for the forward chaining design process.

The permanent improvement of software tools appears today in several programming environments, materialised on object-oriented features, database accessing features and high-interactive user interface. This trend puts ES designer in front of a lot of possible choices and solutions regarding the implementation of an inference mechanism.

Starting from these trends, the next section of the paper will show an implementation of the Structured Query Language (SQL) queries, a technology coming from Database Environments, as a handy possibility of implementing an inference module. The proposed solution works for both forward chaining strategies and backward chaining strategies.

3. A solution for implementing an Inference Mechanism using SQL

Few years ago, the SQL techniques were available only in Database Programming Languages. Today, object programming environments have extensions for using these techniques, making them suitable for

developing database applications, and not only (i.e. for search engines, different AI applications, etc.).

A special instruction of the SQL, named query, retrieves data from a database, constructing a new temporary table, if required. In this way, a search process for locating the records that match a pattern can be easily and optimally done. For indexed databases the query process works very fast.

The syntax for a query is:

```
SELECT    <list of fields>
FROM      <list of tables>
WHERE     <list of conditions>
GROUP BY <list of conditions>
HAVING    <functions>
```

Fig. 2 shows what happens if the following simplified query statement is done.

```
SELECT      Field A AND Field C
FROM        Table 1 AND Table 2
WHERE       Field B="Fact2-B"
```

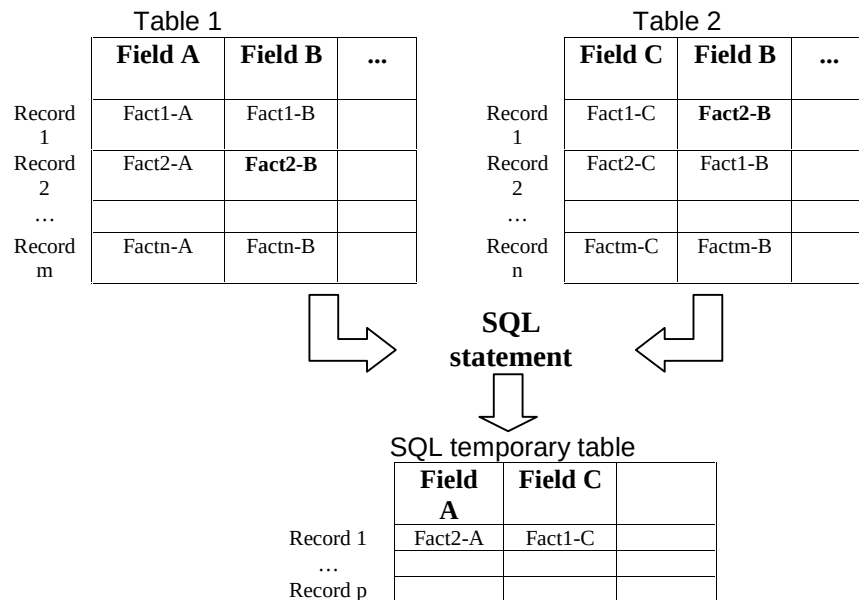


Fig 2. *How a SQL statement is working*

A short discussion about applying the SQL query is given below. The query process is a search process of *Field B* from both *Table 1* and *Table 2*. The searching criteria is that of matching *Field B* records with *Fact2-B* value. When the criteria is accomplished, the corresponding *Field A* contents and *Field C* contents is extracted. Here, record 2 of *Field A* (*Fact2-A*) from *Table 1* is extracted and placed in the SQL temporary table, as well as record 1 of *Field C* (*Fact1-C*) from *Table 2*.

The process goes on until the tables are completely searched. The search process is a **contents search**, every record's contents being examined.

The searching criteria (list of conditions from the *WHERE* query part) can be logical or mathematical (using operators like "<", ">", etc.).

If tables contain knowledge, by using several query constructions (SQL statements), every knowledge piece (fact, rule) can be easily found and extracted, especially if knowledge pieces are unique (making them suitable for table index fields). Therefore, the contents searching process, one of the key problems to set up an ES inference mechanism can be solved.

The inference mechanism shown in fig. 3 can be designed according to the above considerations.

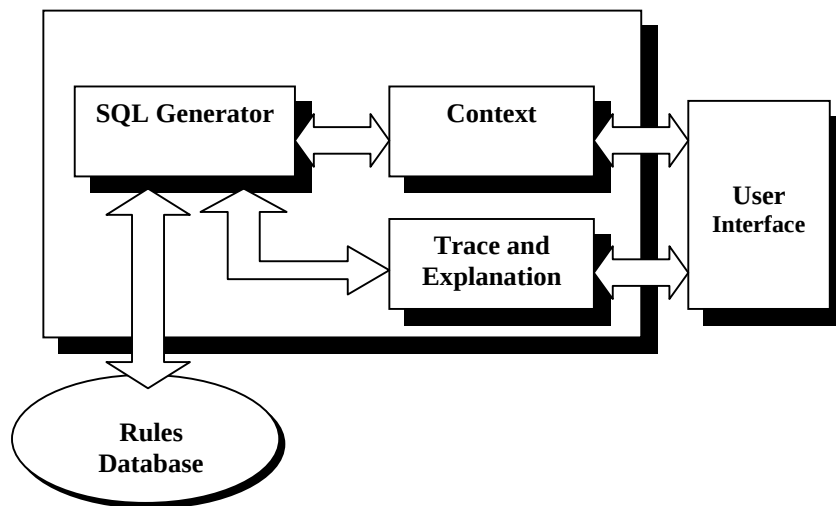


Fig. 3. The designed SQL based inference mechanism for forward chaining inference strategy

An explanation about each module of the SQL based Inference Mechanism is given below.

1. The *Rules Database* is the support for storing rules.
2. The *SQL generator* is responsible for constructing SQL statements and interrogating the *Rules Database*.
3. The *Context* is a dynamic module that stores knowledge existing at the beginning of inference process (as premises), as well as the derived knowledge in the inference process.
4. The *Trace and explanation* module keeps a track of SQL statements, as stages in the inference process.
5. The last two modules have a connection with the *User Interface*, to let him decide about tracing options, setting inference strategy, and so on.

One would see that this technology based on databases offers several advantages:

1. high-speed data storing/retrieval to/from the storing support due to optimisation of search process within a query (especially when the facts are unique and can be indexed);
2. isolation of data path (knowledge) from control path (connection between inference steps), making the ES inference mechanism suitable for almost every type of knowledge;
3. the easiness of constructing complex SQL statements.

Mixing the two strategies is also possible, but some restrictions are to be taken into account.

4. Using the SQL inference mechanism for forward chaining

As stated above, forward chaining inference mechanisms start deriving knowledge from some initial premises (true facts). The inference process stops either when all possible applicable rules were fired or a certain goal was reached.

In the designed system, facts assumed to be true at the start of the inference process are stored in the *Context*. With these facts, a SQL query is done in order to retrieve all rules having the IF part accomplished. One rule is selected and triggered. The THEN part of facts is added as knowledge to the *Context* and the process starts again. To discuss this mechanism, the following rules are considered.

Rule 1

IF
 A is valid
THEN
 B is valid

Rule 2

IF
 B is valid
THEN
 C is valid

Fig. 4 graphically shows how forward chaining goes on. Starting with the premise “A is valid”, the *Context* evolves consequently by applying rule 1 and rule 2. Some facts may lead to opposite conclusions. Thus, an additional verification must be done. Consider, for example, the next rule:

Rule 3

IF
 A is valid
THEN
 B is invalid

Applying this rule will result in storing inside *Context* two opposite facts: *B is valid* gained from rule 2 and *B is invalid* gained from rule 3.

If such conflicts appear, the user can assign priorities to rules. The finally chosen fact inside the *Context* may be that one resulting from a higher priority rule.

5. Using the SQL inference mechanism for backward chaining

The backward chaining mechanisms try to prove a certain goal. In the particular case of SQL based inference mechanism, the goal (or goals) is also stored initially in the *Context*. The systems starts by searching THEN parts of rules that may contain the goal using SQL statements. Then, the IF parts are added to the *Context* and the set of goals is extended. A user dialog can be provided to ask him if the new extracted subgoals are considered known. The process continues until the goal was proved or no more rules are available.

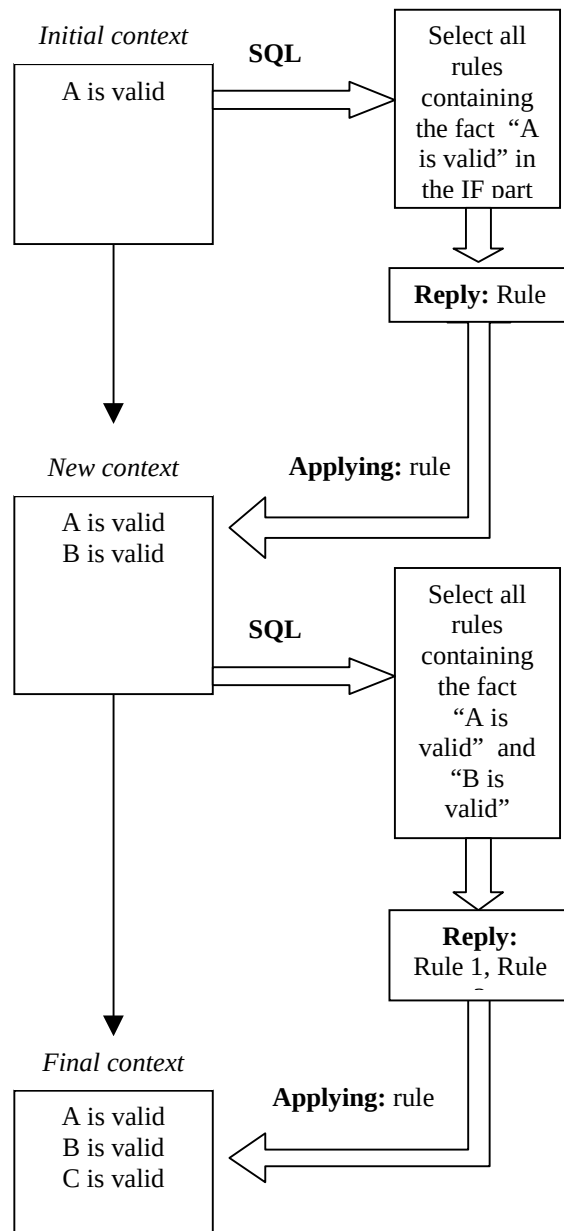


Fig. 4. Forward chaining inference using SQL queries

Having as goal the fact *A is valid*, for the set of rules described above (rule 1 and rule 2), the following steps described by fig. 5 take place.

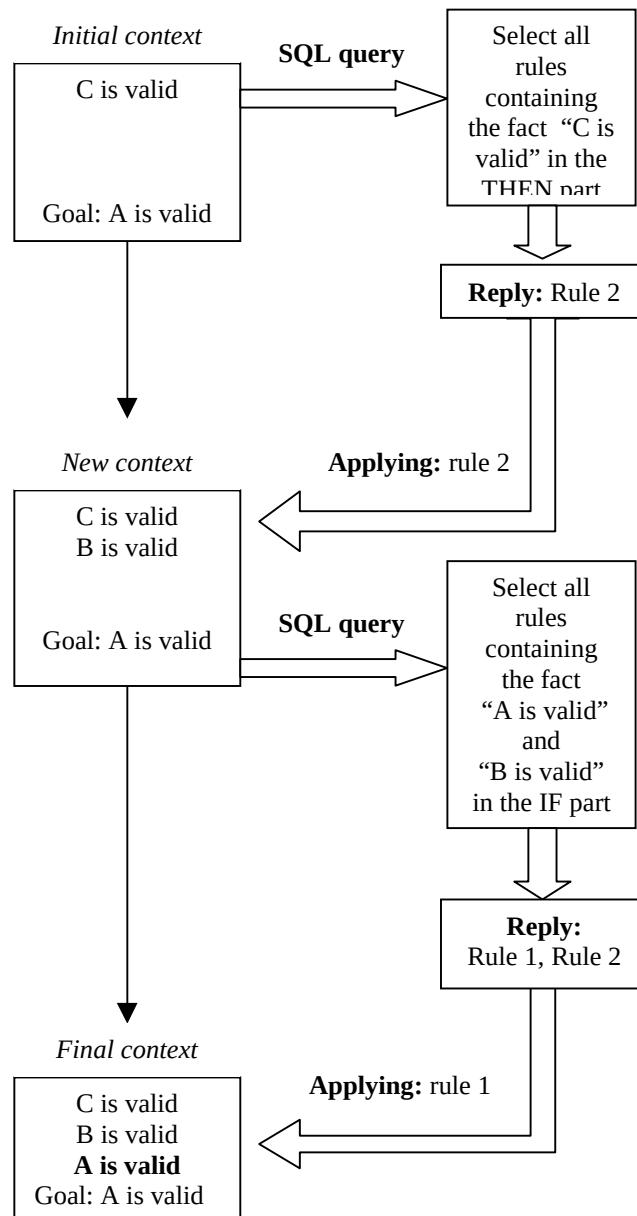


Fig. 5. Backward chaining inference using SQL queries

6. A case study for the implemented ES shell

As shown, both forward and backward chaining strategies can be implemented using the designed SQL based inference mechanism.

The following part gives some implementation issues about a SQL based forward chaining inference mechanism on an ES elaborated by the authors.

The designed ES supports the following data representation formats (Dobre [4], Grigorescu [5]):

- facts (propositions);
- rules (IF-THEN constructions);
- variables (numeric format);
- procedural program calls for computational tasks.

For mapping these data structures in a database appropriate format, a form like that appearing in the fig. 6 was chosen.

One would see that the represented fields are not necessarily same table fields, but may be found in different tables, among with a precisely defined table relationship. Here, the *Rules table* is used for storing rules and *Facts table* for storing facts. The *Fact number* field is provided for easily adapting facts to SQL statements. A fact may appear several times in rules so using this additional field reduces also the amount of data written in *Facts table*. The *Context Table* is provided for storing dynamic knowledge derived in each inference step (i.e. a SQL query).

For implementing an inference chain, each inference step does the following actions:

1. a *Selection process* is executed starting with the *Context* available facts and taking into account the SQL reserved fields;
2. SQL query is executed in a *Query process*; as a result, a new subset of possible applicable rules is retrieved;
3. according to a strategy and in concordance with SQL fields, the applicable rule(s) is selected; the *Context* is updated with new derived facts;
4. the SQL fields for the selected rule(s) and retrieved facts are modified accordingly in an *Update process*.

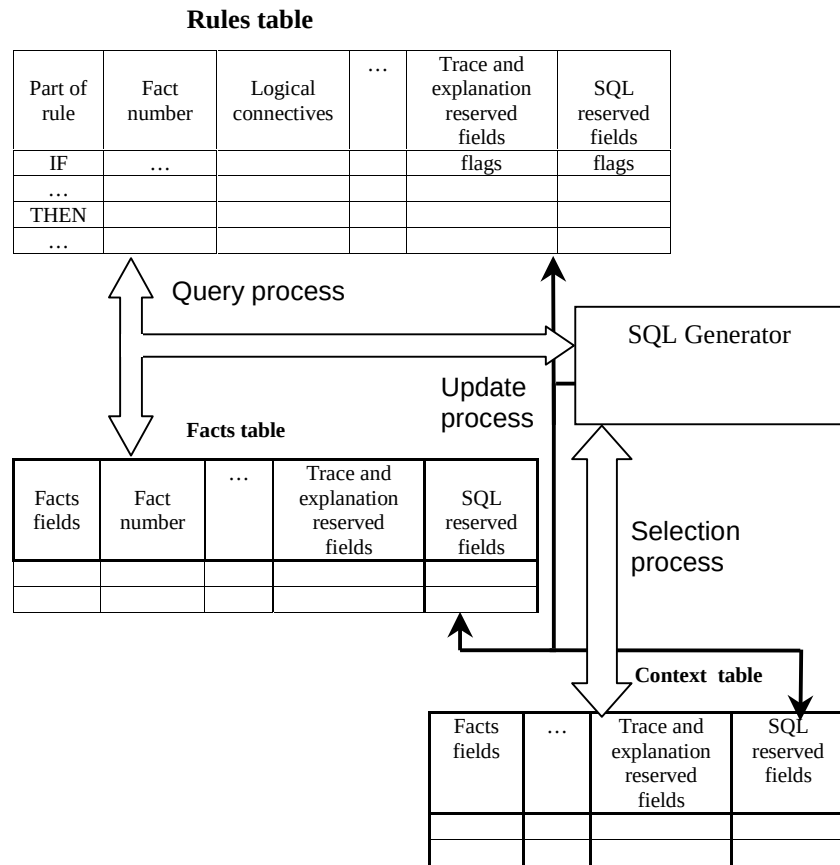


Fig. 6. Data structure using database formats

The actions described above may be recorded within the *Tracing and explanation* module (see fig. 3), if the user wants to.

The ES user interface that implements this inference mechanism is shown in fig. 7 and 8. The user inputs the initial true facts and variables values, if required (fig. 7), creating the initial *Context*.

After these premises were pointed out, the inference mechanism of the designed ES may be triggered in two different ways:

1. a step by step operating mode;
2. an automatic operating mode.

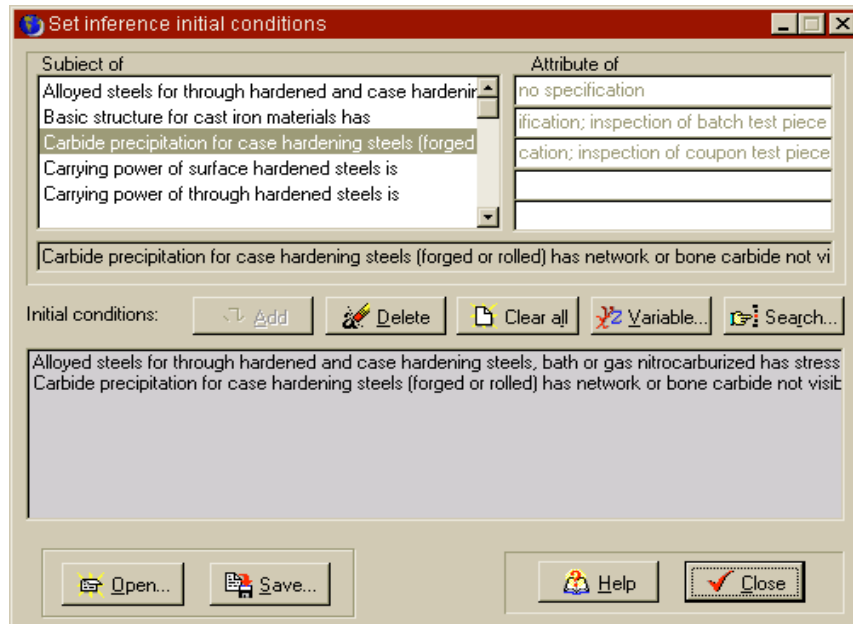


Fig. 7. *Setting up the inference premises*

The **step by step mode** waits for user decision after each inference step (i.e. SQL query is done); this can be very useful when debugging the system. The **automatic operating mode** is the casual working way of the system. A screen showing an user interface after an inference process took place is provided in fig. 8.

7. Conclusions

- The use of the SQL technology in the ES as AI tool is an actual and powerful possibility of elaborating inference mechanisms due to specific advantages.
- The integration of SQL technologies within powerful programming environments makes them suitable for developing ES shells.
- The SQL technology consists in complex search statements in large volume databases, and can be successfully applied in inference mechanisms.

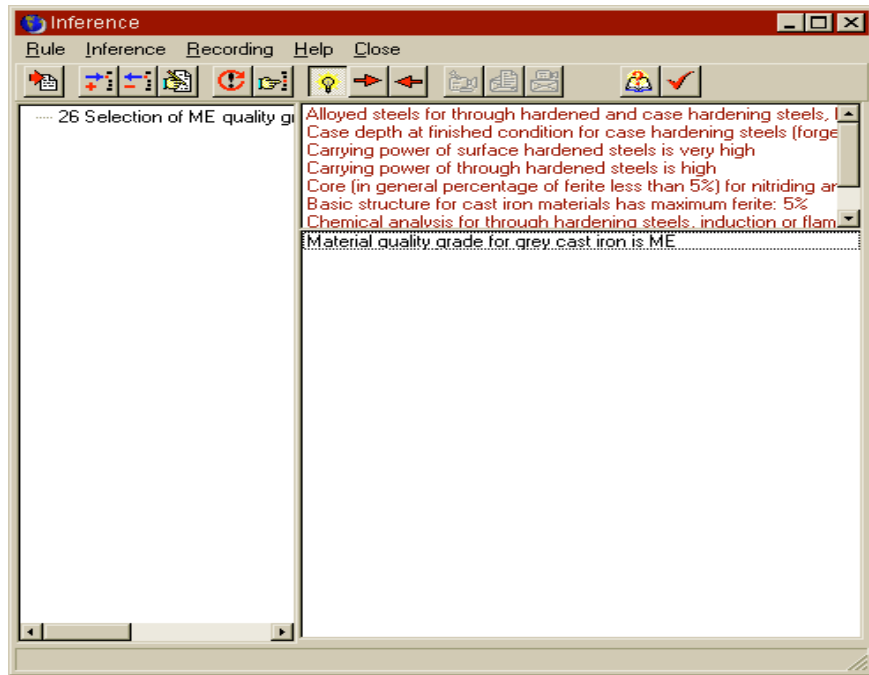


Fig. 8. User screen after the inference mechanism was triggered

- The described way of SQL implemented technology brings new advantages in building an ES shell environment.
 1. high-speed data storing/retrieval to/from the storing support due to optimisation of search process within a query;
 2. isolation of data path (knowledge) from control path (connection between inference steps), making the ES inference mechanism suitable for almost every type of knowledge;
 3. easiness of constructing complex SQL statements.
- An ES shell elaborated by authors implemented using SQL technologies is briefly described in the paper.

References

1. **COYNE R.,D., ET AL.**, *Knowledge-Based Design Systems*, Addison Wesley 1990.
2. **CARSTOIU, D.**, *Sisteme expert*. ALL, Bucuresti, 1994.
3. **DURKIN, J.** *Expert Systems*. Prentice Hall, New Jersey, 1995.
4. **DOBRE, G., RADULESCU, G., CARSTOIU, D., GRIGORESCU, C.** *Intelligent Environment for Optimising of the Mechanical Engineering Design*. International Computer Science Conference microCAD'97, Miskolc, 1997, 41-46.
5. **GRIGORESCU, C., DOBRE, G., PICASSO, B., RADULESCU, G., CARSTOIU, D.** *An Intelligent Environment for Mechanical Engineering Design. Application to Gear Transmissions*. Twelfth International Conference Applications of Artificial Intelligence in Engineering AIENG 97, Capri, Italy, Computational Mechanics Publications, 1997.