THE SIMULATION OF BUSINESS RULES IN ACTIVE DATABASES USING EXPERT SYSTEM APPROACH

Ivan Bruha, Frantisek Franek (McMaster University), and Vladimir L. Rosicky (Terren Corp.)

- AUGMENTATION OF DBMS WITH RULES TO SAFEGUARD THE INTEGRITY OF DATA IN THE DATABASE

- TWO BASIC CATEGORIES OF RULES

  1. PASIVE RULES – ALSO CALLED INTEGRITY CONSTRAINTS – THEY NEVER CAUSE A MODIFICATION OF THE CONTENTS OF THE DATABASE (HENCE "PASIVE" RULES), JUST INDICATE WHEN A CONSTRAINT IS VIOLATED. PASIVE RULES MOSTLY STATED IN A DECLARATIVE LANGUAGE (E.G. REFERENCE INTEGRITY, NULL VALUES, DUPLICATE VALUES ETC.)

2. ACTIVE RULES – ALSO CALLED TRIGGERS – VERY OFTEN USED TO REPAIR CONSTRAINT VIOLATIONS OR TO EMULATE ADDITIONAL CONSTRAINTS. THEY MODIFY THE CONTENTS OF THE DATABASE (HENCE “ACTIVE” RULES). MOSTLY EXPRESSED IN A PROCEDURAL LANGUAGE.

- RECENT RESEARCH TOPIC IN THE AREA OF ACTIVE RULES – BUSINESS RULES – THEY ARE MENT TO SAFEGUARD THE APPLICATION SPECIFIC (BUSINESS) INTEGRITY OF THE DATA IN THE DATABASE.

- BUSINESS RULES MOSTLY EXPRESSED IN THE SAME PROCEDURAL LANGUAGE AS ALL OTHER ACTIVE RULES.

- AIM OF OUR RESEARCH:

1. EMULATE BUSINESS RULES ON TOP OF A GIVEN RELATIONAL DATABA-SE.

2. EXPERIMENT WITH BUSINESS RULES EXPRESSED MOSTLY IN A DECLARATIVE LANGAUGE.

THIS WAS ACHIEVED USING THE FOLLOWING SETUP (FRAMEWORK)

REGULAR QUERIES THROUGH A TRIG-GER INVOKE McESE TO PROCESS A PARTICULAR RULESET.

FIG 1

McESE RULES – DECLARATIVE, ORDER NOT RELEVANT, STRATIFIED, NO "CYCLES" (NO LOOPS OR RECURSION).

r1: 0.3*P1("abc")[>=.3] & ~P2(x,y) & P3(2,3.4,y)
= f =>
S(y,x)[>.6]

SQL-AUGMENTED McESE RULES – RESULTS OF SQL QUERIES CAN BE USED AS VALUES, WHILE THE QUERIES THEMSELVES CAN BE USED AS PREDICATES.

PROCEDURAL ASPECTS OF McESE RULES:

1. CVPF's: Certainty Value Propagation functions (**f** above)
2. Atomic predicates (not occuring on the righ-hand-side of any rule) – correspond to built-in procedures with paramaters.

THUS, PROCEDURAL ASPECTS OF BUSINESS RULES DEFERRED TO THE LOWEST LEVEL POSSIBLE.

CASE STUDY – AUTOMATIC INVENTORY RELEASE.

A travel agency may purchase a block of airline tickets for a particular flight and if they are not sold by 30 days prior to the flight, the allotment of tickets that can be sold, is reduced by 50% and thus the travel agency can sell fewer tickets and all additional sales must be done through the airline; 5 days prior to the flight, the whole allotment of tickets that can be sold is reduced to 0 and thus the travel agency cannot sell any more tickets on its own and all sales must be done through the airline.

table *inventory_block* has attributes

- **BlockId** (unique block identifier)
- **SupplyId** (unique supply identifier)
- **Date** (date for which the inventory in the block is destined)
- **Status** (there are three possible values: ***FREE*** indicating that any unsold quantity from the inventory block can be freely sold, ***REQST*** indicating that any sale must go through a request to the supplier, and finally ***STOP*** indicating no further sale is possible)
- **Allotment** (the original quantity)
- **Unsold** (unsold portion of the original allotment).

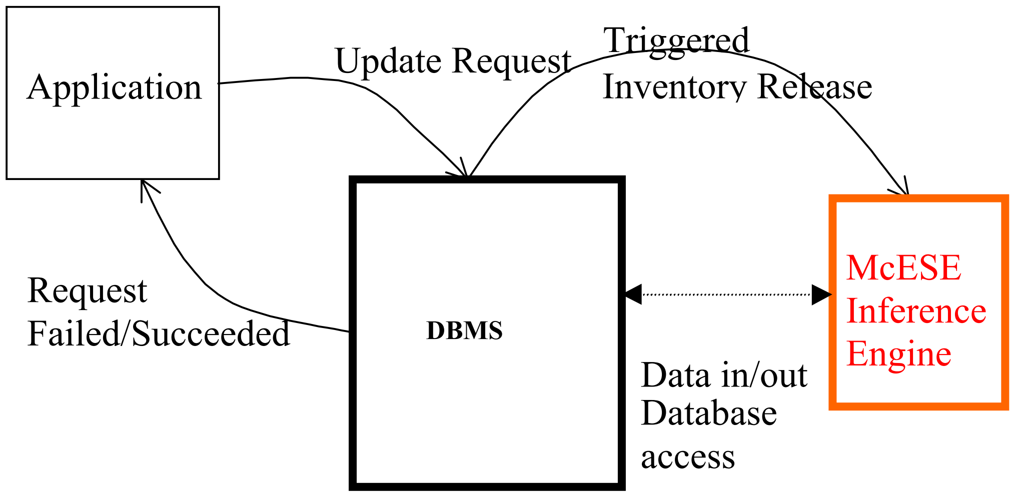
Rule1: 20 days prior **Date**, reduce **Unsold** by 50%
Rule2: 10 days prior **Date** change **Status** to ***REQST*** and reduce **Unsold** by 50%
Rule3: 2 days prior **Date** change **Status** to ***STOP***

```
DEF %date=(SELECT Date
   FROM inventory_block
   WHERE SupplyId=%supply_id)

DEF %status=(SELECT Status
  FROM inventory_block
  WHERE SupplyId=%supply_id)

R1:%date<=%today+20 &
%status=FREE & ~R2 & ~R3
        ==>
(UPDATE inventory_block
 SET Unsold=Unsold*0.5
 WHERE SupplyId=%supply_id)
```

```
R2: %date<=%today+10 &
%status=FREE & ~R1 & ~R3
        ==>
(UPDATE inventory_block
 SET Unsold=Unsold*0.5,
 Status=REQST
 WHERE SupplyId=%supply_id)

R3:%date<=%today+2 & ~R1 & ~R2
        ==>
(UPDATE inventory_block
 SET Status=STOP
 WHERE SupplyId=%supply_id)
```

CONCLUSION

The above described case study of automatic inventory block release indicates, despite the simplicity with which it was presented here, that SQL-augmented expert systems like McESE could be successfully utilized as tools for design of active business rules. The main contribution of this approach is the declarative aspect brought to the design of business rules which facilitates understanding and easier development. The additional contribution can be seen in the processing of uncertainty which is very often a part of business activities and cannot be accommodated by the traditional business rules in active databases.


Application
Update Request
Triggered
Inventory Release
DBMS
McESE
Inference
Engine
Request
Failed/Succeeded
Data in/out
Database
access