Forms : réalisation d'un moteur d'inférence

Objectif

Ecrire un moteur d'inférence propositionnel dont la stratégie est la suivante :
chaînage avant, exécution de toutes les règles déclenchables à chaque cycle, exécution jusqu'à saturation des règles.

Ce moteur manipule des règles de la forme suivante :

```
SI 'beautemps' ET 'pas de travail' ABORS 'promenade'
```

Selon l'exemple ci-dessus, il doit être capable de déduire (inférer) le fait 'promenade' si les faits 'beautemps' et 'pas de travail' sont vrais. Les règles sont stockées dans les tables *regles* et *conditions*. La table *base_de_faits* contiendra les faits préétablis ainsi que les faits déduits par le moteur au fur et à mesure de l'exécution des cycles.

Ainsi, le schéma de la base de données utilisée pour stocker les règles et les faits est le suivant :

```
REGLES (NR, ACTION, GELE)
CONDITIONS (NR*, CONDITION)
BASE_DE_FAITS (FAIT : action, NR*)
```

Pour tester le moteur, utiliser l'extension suivante :

```
select * from regles
NR         ACTION     GELE       PRIORITE
---------- ---------- ---------- ----------
         1 d                   0
         2 f                   0
         3 e                   0
         4 g                   0
         5 g                   0

select * from conditions
NR         CONDITION
---------- ----------
         1 a
         1 b
         1 c
         2 d
         2 e
         3 b
         4 f
         5 d
         5 b

select * from BASE_DE_FAITS
FAIT       NR
---------- ----------
a
b
c
```

Les tables *règles* et *conditions* de cette extension correspondent aux règles suivantes :

```
1: SI a, b, c ALORS d
2: SI d, e ALORS f
3: SI b ALORS e
4: SI f ALORS g
5: SI d, b ALORS g
```

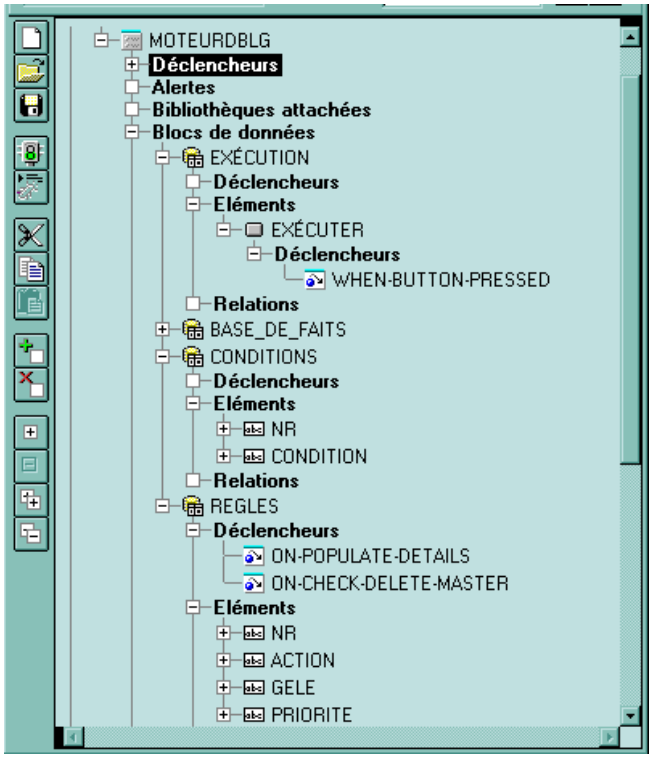
Au départ, la table faits contient a, b, c de sorte que seules les règles 1 et 3 sont exécutables.

A l'exécution du premier cycle, la règle 1 ajoute le fait d et la règle 3 ajoute le fait e. Après l'exécution de ce cycle, la règle 2 devient exécutable. Le moteur va ainsi procéder par cycles jusqu'à ce qu'il n'y ait plus de règles exécutable.

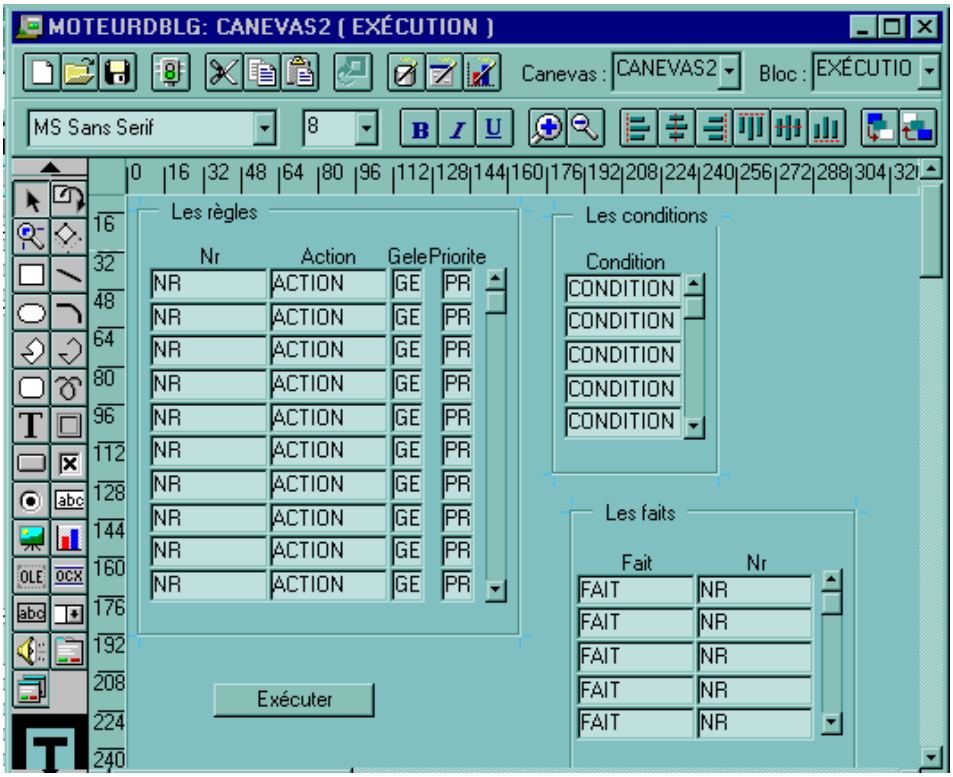
Remarque : une fois exécutées, les règles sont gelées de sorte à n'être exécutées qu'une fois. Les priorités ne sont pas utilisées dans cet exemple.

Réalisation sous Oracle

Création d'un module sous développer2000. Ce module comporte 4 blocs : le bloc EXECUTION est un bloc non basé comportant un bouton chargé d'exécuter le code du moteur ; le bloc BASE_DE_FAIT est basé sur la table représentant la base de faits, il est indépendant des autres blocs ; les blocs CONDITIONS et REGLES sont respectivement basés sur les tables contenant la lite des conditions de toutes les règles et la liste des règles, il sont reliés par une relation maître-détail basée sur la condition de jointure regles.NR=conditions.NR, le bloc maître étant REGLES.



Le canevas permettant d'afficher les items de ces blocs est le suivant :



Le code du déclencheur associé au bouton exécuter est le suivant :

```
DECLARE
   CURSOR c1 is
      SELECT NR, action FROM regles
         WHERE gele = 0  and
               not exists(SELECT condition
                          FROM conditions
                          WHERE conditions.NR = regles.NR and
                                not exists(SELECT fait
                                           FROM BASE_DE_FAITS
                                           WHERE fait = condition
                                           ))
      FOR UPDATE OF gele;

      my_NR   NUMBER;
      my_action regles.action%TYPE;
      i NUMBER;
      fait_existant NUMBER;

BEGIN
bell;
  OPEN c1;
  i:=1;
  FETCH c1 INTO my_NR, my_action;
  IF c1%NOTFOUND THEN
    message('aucune règle n''est exécutable');
  END IF;
  WHILE c1%FOUND LOOP   -- tant qu'il y a des règles exécutables
    message('cycle '||i);
   -- Exécution d'un cycle
   WHILE c1%FOUND LOOP  -- exécution de toutes les règles déclenchables
      message('exécution de la règle : '||my_NR);
      SELECT count(fait) INTO fait_existant
       FROM BASE_DE_FAITS WHERE fait = my_action;
      IF fait_existant = 0 THEN
          INSERT INTO BASE_DE_FAITS VALUES (my_action, my_NR);
      ELSE
          bell;
          message('tentative d''insertion d''un fait déjà existant par la règle '||my_NR);
      END IF;

      UPDATE regles SET gele = 1
             WHERE CURRENT OF c1;
      FETCH c1 INTO my_NR, my_action;
    END LOOP;
    CLOSE c1;
    --COMMIT; éventuelle validation après chaque cycle
    Go_block ('BASE_DE_FAITS'); -- rafraichissement de l'affichage des faits
    EXECUTE_QUERY;
    Go_block ('REGLES');  -- rafraichissement de l'affichage des règles
    EXECUTE_QUERY;
    OPEN c1;       -- recherche des nouvelles règles exécutables
    i:=i+1;
    FETCH c1 INTO my_NR, my_action;
  END LOOP;
  CLOSE c1;
  COMMIT;
END;
```