Pièges à éviter dans l'écriture de Règles de Gestion Adélia.

- Les articles -

Date de mise en ligne : mardi 18 mai 2004

Description :

Quand on crée des règles de gestion avec déclaration de vue en interne, il y a quelques pièges à éviter.

Environnement iSeries

Quand on crée des règles de gestion avec déclaration de vue en interne, il y a quelques pièges à éviter. Notez que ces pièges sont identiques si on travaille avec des macro-instructions.

Cas n° 1 :

supposons une vue ARTICLE déclarée dans la règle de gestion RECUP_LIB_ART comme suit :

```
IDENTIFIER ARTICLE ARTICLL1 CLE ;:01 ;:02
```

Le cas n° 1 est le pire qu'on puisse rencontrer car, si la règle de gestion est déclarée plusieurs fois dans un même programme avec des variables passées en paramètre qui ne sont pas les mêmes à chaque appel, alors c'est le premier appel de la règle de gestion qui " gagne " puisque la vue sera déclarée dans le programme (la compilation du programme ne posant pas de problème dans ce cas) avec les variables utilisées pour le premier appel de la règle de gestion. Dans l'exemple ci-dessous, c'est la variable Y_COD_ART qui gagne sur la variable ZW_COD_ART :

```
RECUP_LIB_ART(W_COD_SOC ; Y_COD_ART)
RECUP_LIB_ART(W_COD_SOC ; ZW_COD_ART)
```

Cas n° 2 :

supposons une vue ARTICLE déclarée dans une règle de gestion RECUP_LIB_ART comme suit :

```
IDENTIFIER ARTICLE ARTICLL1 CLE ;W_COD_SOC ;W_COD_ART
```

supposons une vue ARTICLE déclarée dans une règle de gestion EXIST_COD_ART comme suit :

```
IDENTIFIER ARTICLE ARTICLL1 CLE ;ZW_COD_SOC ;ZW_COD_ART
```

Supposons maintenant que ces 2 règles de gestion RECUP_LIB_ART (récupération du libellé d'un article) et EXIST_COD_ART (contrôle de l'existence d'un article) soient appelées dans le même programme. A la compilation du programme, c'est la première règle de gestion appelée qui " gagne " sur l'autre. En effet, la compilation du programme ne plantera pas et ne signalera même pas un conflit dans la déclaration de la vue RG_ARTICLL1 entre les 2 règles de gestion. Par contre, à l'exécution, la lecture sur la vue RG_ARTICLL1 se fera toujours avec la même clé, quelle que soit la règle de gestion appelée à un moment donné.

Cas n° 3 :

supposons une vue ARTICLE déclarée dans un programme ADELIA et pointant sur le fichier logique ARTICLL1 avec comme zones de clé (W_COD_SOC et W_COD_ART)

supposons une vue ARTICLE déclarée dans une règle de gestion RECUP_LIB_ART comme suit :

```
IDENTIFIER ARTICLE ARTICLL1 CLE ;ZW_COD_SOC ;Y_COD_ART
```

A la compilation du programme, c'est la vue déclarée dans le programme qui " gagnera " sur la vue déclarée dans la RG. Là encore, la compilation du programme ne plantera pas et ne signalera même pas un conflit dans la déclaration de la vue RG_ARTICLL1 entre les 2 règles de gestion. Par contre, à l'exécution, la lecture sur la vue RG_ARTICLL1 se fera toujours avec la même clé, quelle que soit la règle de gestion appelée à un moment donné

Bonnes pratiques :

- veiller à utiliser un nom de vue de préférence unique, ou en tout cas ne risquant pas d'être utilisé à l'intérieur d'un programme. Par exemple, les noms de vue déclarées dans les règles de gestion pourront être préfixés par RG et les noms de vue déclarées dans les macro-instructions pourront être préfixés par MI.

- Déclarer de préférence une vue dans une RG avec ses noms de propriétés logiques et alimenter ces propriétés logiques avant les ordres de lecture avec les variables reçues en paramètre.

- Si le fichier déclaré dans une RG est susceptible d'être déclaré en mise à jour dans un programme utilisant cette RG, ou d'être utilisé en mise à jour dans une autre RG utilisée dans le même programme, il est recommandé de lire systématiquement l'enregistrement avec l'instruction de déverrouillage d'enregistrement *UNLCK de façon à éviter les verrouillages indésirables.

Exemples de bonnes pratiques : soit 2 règles de gestion RECUP_LIB_ART et EXIST_COD_ART utilisant le même fichier logique et susceptibles d'être utilisées dans les mêmes programmes.

Règle de gestion EXIST_COD_ART : contrôle d'existence d'un article

```
CLASSE : Paramètre

 PARAMETRES :

 N° CODE               FORMAT   CONTROLE
 1  Code société          3     OBLI
 2  Code article          8 0   OBLI
 3  Libellé article      30

 SOURCE :

 IDENTIFIER RG_ARTICLL1 ARTICLL1 CLE ;S3_COD_SOC ;S3_COD_ART
 S3_COD_SOC      = :01
 S3_COD_ART      = :02
 :03 = *BLANK
 LIRE RG_ARTICLL1 *UNLCK
 SI   RG_ARTICLL1 N_EXISTE_PAS
 PREPARER_MSG MSG0337 :02
 ANOMALIE
 SINON
 :03  = S3_LIB_ART
 FIN

 Règle de gestion RECUP_LIB_TOU : récupération du libellé d'un article
```

```
CLASSE : Paramètre

PARAMETRES :

N° CODE               FORMAT   CONTROLE
1  Code société          3     OBLI
2  Code article          8 0   OBLI
3  Libellé article      30

SOURCE :

IDENTIFIER RG_ARTICLL1 ARTICLL1 CLE ;S3_COD_SOC ;S3_COD_ART
S3_COD_SOC      = :01
S3_COD_ART      = :02
:03 = *BLANK
LIRE RG_ARTICLL1 *UNLCK
SI   RG_ARTICLL1 EXISTE
:03 = S3_LIB_TOU
FIN
```