Du bon usage des listes mémoires

- Les articles -

Date de mise en ligne : mardi 20 décembre 2005

Description :

Le concept de "liste mémoire" a été introduit par Hardis, dans la version 8 d'Adelia. Cet article a pour objectif de vous donner un aperçu des avantages et inconvénients de cette technique.

Environnement iSeries

Le concept de liste mémoire est une véritable bouffée d'oxygène pour les développeurs Adelia, qui bénéficient ainsi d'une fonctionnalité très puissante, pouvant remplacer avantageusement le système des fichiers temporaires ("montés" dans QTEMP) cher aux développeurs AS/400. Mais l'on verra que tout n'est pas rose avec cette nouvelle technique, et qu'il vaut mieux en connaitre les inconvénients pour ne pas l'utiliser de manière inconsidérée.

Entrons tout de suite dans le vif du sujet avec un petit programme exemple :

```
*-- déclaration liste mémoire LST_ART, mixant la structure du fichier détail de commande (CDCLDE) avec des zones supplémentaires telles que "libellé article" et "adresse logistique" provenant du fichier ARTICLE

 DECL LISTE LST_ART *REF_F(CDCLDE) A1_LIB_ART A1_ADR_LOG

 *-- Lecture du fichier détail de commande et alimentation de la liste mémoire

 POSITIONNER_AV CDCLDE
 LIRE_SUIVANT   CDCLDE
 TANT_QUE       CDCLDE EXISTE
 LIRE ARTICLE
 SI   ARTICLE EXISTE
 *-- ajout d'un poste dans la liste mémoire
 INSERER_ELT LST_ART
 FIN
 LIRE_SUIVANT CDCLDE
 REFAIRE

 *-- Tri des articles de la liste mémoire sur l'adresse logistique et le code article (du fichier CDCLDE)

 TRIER_LST LST_ART A1_ADR_LOG K2_COD_ART

 *-- Lecture séquentielle de la liste mémoire pour édition des articles retriés (l'ordre LIRE_P_ELT permet de se positionner sur le premier poste de la liste mémoire triée)

 LIRE_P_ELT LST_ART
 W_COD_RET_LST = &CODE_LST(LST_ART)
 TANT_QUE W_COD_RET_LST = *NORMAL
 *-- alimentation des zones de la maquette et édition
 ZW_COD_ART = K2_COD_ART
 ZW_ADR_LOG = A1_ADR_LOG
 ZW_LIB_ART = A1_LIB_ART
 *-- lecture du poste suivant de la liste mémoire
 LIRE_AV_ELT LST_W
 W_COD_RET_LST = &CODE_LST(LST_W)
 REFAIRE

 etc...
```

J'ai choisi un exemple simple, mais fonctionnel, qui je l'espère vous permet de bien comprendre l'utilisation qui est

faite ici de la liste mémoire.

Je trouve que ce type d'utilisation de la liste mémoire est réellement avantageux. Dans l'exemple ci-dessus, on crée une structure de données temporaire (la liste mémoire LST_ART), en mixant la structure du fichier détail de commande (fichier CDCLDE) avec des zones supplémentaires provenant du fichier article. On aurait d'ailleurs pu ajouter à la structure de la liste mémoire des variables de travail n'appartenant à aucun fichier. On peut aussi constituer une liste mémoire **ne s'appuyant que** sur des variables de travail et aucune structure de fichier. Ensuite, on alimente cette liste mémoire avec les lignes détail d'une commande, on la retrie selon des critères qui n'existaient pas dans le fichier détail de commande (tels que l'adresse logistique), et on édite les articles tel que l'utilisateur nous l'a demandé. Que demande le peuple ?

Dans quels cas utiliser les listes mémoires ?

- chaque fois qu'on est tenté de créer, dans un programme batch, un fichier temporaire ("monté" dans QTEMP), dans un contexte proche de celui présenté dans l'exemple ci-dessus.

- chaque fois que dans un programme interactif, vous avez besoin d'afficher des données dans un sous-fichier, éventuellement en croisant des données provenant de plusieurs fichiers, avec éventuellement plusieurs critères de tris, et à condition que le nombre d'enregistrements à afficher ne dépasse pas une centaine d'enregistrements (je propose ce chiffre, mais il vaut mieux l'apprécier selon la puissance de votre ou de vos AS/400, par la réalisation de tests adaptés).

Dans quels cas ne pas utiliser les listes mémoires ?

- si le volume de données à traiter au sein de la liste mémoire menace de dépasser la centaine d'enregistrements, je recommande vivement de faire des tests pour vous assurer que le niveau de performance est acceptable pour vous. J'ai vu un AS/400 de production plonger littéralement, en essayant de trier une liste mémoire contenant plus d'un millier d'enregistrements, alors qu'un autre AS/400 (beaucoup plus puissant), sur un volume de données équivalent, effectuait son travail de tri dans un délai tout à fait acceptable (pour un traitement batch).

- si vous avez besoin d'accéder à des postes précis de la liste mémoire. En effet, il n'existe pas d'ordre Adelia équivalent d'un "LIRE" (ou "CHAIN" en RPG), pour les listes mémoires. Elles sont donc bien adaptées pour une lecture séquentielle de données, mais pas pour un accès direct, car on est obligé de balayer séquentiellement la liste mémoire pour trouver l'enregistrement adéquat. J'avais voulu utiliser cette technique dans un programme interactif, en offrant à l'utilisateur la possibilité de modifier le tri des données du sous-fichier au moyen d'une touche de fonction. D'ailleurs ça fonctionnait très bien, l'utilisateur était ravi, et moi aussi, mais j'avais besoin dans ce même programme que l'utilisateur puisse modifier certaines données dans le sous-fichier. Or, chaque modification impliquait de rebalayer la liste mémoire depuis le début pour identifier l'élément à modifier. Les performances se sont révélées très mauvaises, dès que la liste mémoire a dépassé une cinquantaine d'enregistrements (avec de grosses différences, il est vrai, d'un AS/400 à l'autre). Pour couper court à ces problèmes de performances, j'ai renoncé à utiliser cette technique dans ce cas précis, et j'ai dû me résoudre à revenir à un fichier de travail temporaire.

En conclusion : j'espère que ce court article vous aidera à évaluer l'intérêt des listes mémoires dans votre contexte applicatif, et qu'il vous donnera envie de les utiliser. Pour ma part, je suis confronté à un parc d'AS/400 hétérogène, dont les puissances varient du simple au quintuple, et je suis obligé de jouer la sécurité, en développant des programmes qui ne souffrent pas de défauts de performances d'une machine à l'autre. C'est pourquoi j'ai banni l'utilisation des listes mémoire dans certains contextes evoqués ci-dessus, sans pour autant l'écarter complètement. Je l'utilise par exemple avec bonheur dans des programmes batchs tels que celui présenté en exemple. Je ne peux

que vous recommander de tester cette fonctionnalité, car malgré ses limites, Hardis nous a fait un beau cadeau en l'intégrant à Adelia/400, et il serait dommage de s'en priver.

Pour information, vous trouverez ci-dessous la liste des principales commandes liées aux listes mémoires. Leur utilisation est bien expliquée dans l'aide d'Adelia.

```
CHARGER_SQL_LST     Charger une liste par SQL
 COPIER_LST          Copie une liste
 INSERER_ELT         Insère un élément
 INSERER_LST         Insère une liste
 LIRE_AR_ELT         Lit l'élément précédent
 LIRE_AV_ELT         Lit l'élément suivant
 LIRE_D_ELT          Lit le dernier élément
 LIRE_P_ELT          Lit le premier élément
 MODIFIER_ELT        Modifie un élément
 SUPPRIMER_ELT       Supprime un élément
 TRIER_LST           Trier une liste
 VIDER_LST           Vide une liste
```