Modifier la structure d'un fichier (CHGPF).

- Les articles -

Date de mise en ligne : mercredi 11 août 2004

Date de parution : 4 novembre 2003

Description :

Comment gérer l'impact d'une modification dans la structure d'un fichier, sur les applications en production.

Environnement iSeries

Au cours du cycle de vie d'un logiciel, il n'est pas rare de devoir modifier la structure d'un fichier pour ajouter de nouvelles colonnes (les développeurs RPG emploient plus volontiers le terme de "zones", par contre les développeurs SQL préfèreront le terme de "colonne"), ou pour rallonger une colonne numérique ou alphabétique taillée "trop juste" lors de la conception initiale du logiciel.

Gérer les modifications de structure

Lorsqu'on modifie la structure d'un fichier, cela a un impact sur tous les programmes exploitant ce fichier. En effet, si le fichier n'est pas au même niveau que les programmes, ces derniers "planteront" à l'exécution, avec toutes les conséquences fâcheuses que l'on connaît (allant de "pas grave" à "base de donnée corrompue"). Il est possible de tricher en modifiant le fichier au moyen de la commande CHGPF en mettant le paramètre LVLCHK (contrôle de niveau) à *NO, mais cette solution est dangereuse. On peut rencontrer plusieurs cas :

1. pas grave : les programmes lisent le fichier sans effectuer de mise à jour, rien à craindre, tout devrait bien se passer
2. pas grave non plus : certains programmes font des mises à jour (ou création) dans le fichier modifié et la colonne ajoutée dans ce fichier est **alphanumérique**. Pour les programmes non recompilés après la modification du fichier, lors de leur exécution, la colonne ajoutée au fichier modifié n'existe tout simplement pas, il n'y a pas de conséquence fâcheuse à craindre et tout devrait bien se passer.
3. grave (voire très grave) : certains programmes font des mises à jour (ou création) dans le fichier modifié et la colonne ajoutée dans ce fichier est **numérique**. Dans ce cas, les programmes non recompilés après la modification du fichier ne connaissent pas la colonne ajoutée et vont se comporter comme si elle n'existait pas. Cela aura pour conséquence qu'on va se retrouver avec des colonnes numériques contenant des blancs. Query n'aime pas les colonnes numériques contenant des caractères blancs ou alphabétiques, et les programmes RPG qui rencontrent ce genre de colonne non plus (ils plantent en indiquant une erreur de donnée numérique).

Pour éviter les mauvaises surprises, certains chefs de projet (dont je suis), préfèrent ne pas toucher le paramètre LVLCHK du fichier modifié, et recompiler tous les programmes utilisant ce fichier, qu'ils utilisent le fichier en lecture ou en mise à jour. Bien sûr, on peut rencontrer des situations où la modification du paramètre LVLCHK peut être envisagée, mais pour ma part, je préfère l'écarter aussi souvent que possible.

Modifier un fichier n'est pas le plus compliqué, on modifie sa structure par un fichier DDS que l'on recompile (ainsi que ses fichiers logiques qu'on aura pris soin de supprimer au préalable), ou par SQL. Ensuite on recompile tous les programmes utilisant ce fichier. Attention, ne vous amusez JAMAIS à changer le type d'une colonne existante (par exemple d'alphanumérique à numérique) sinon vous allez au devant de graves ennuis lors de la phase de récupération des données de l'ancien fichier vers le nouveau.

Comment retrouver à coup sûr l'ensemble des programmes utilisant le fichier modifié : il n'y a pas de réponse unique, tout dépend des outils de développement que vous utilisez, et de votre méthodologie. Mais si vous avez des doutes, vous pouvez vous appuyer sur un outil comme la commande AS/400 DSPPGMREF qui permet d'extraire des informations précieuses pour la maintenance d'une application (cf. page correspondante en cliquant sur le lien). Mais est-on sûr de disposer de tous les objets susceptibles d'attaquer le fichier ? N'y a-t-il pas sur l'AS/400 destinataire des programmes développés spécifiquement pour lesquels vous ne disposez pas des sources ? Si c'est le cas, vous avez tout intérêt à prévenir les utilisateurs de l'AS/400 destinataire qu'ils devront remettre à niveau leurs programmes spécifiques après votre mise à niveau, ou alors, faute d'interlocuteur (si vous travaillez sur une application ancienne qui n'est plus maintenue) vous pouvez vous rabattre sur la solution LVLCHK(*NO), et prier...

Plus sérieusement, vous pouvez employer la commande DSPPGMREF également sur ces programmes spécifiques et ainsi repérer les programmes susceptibles de faire des mises à jour sur le fichier physique ou l'un de ses fichiers logiques.

Vous noterez que je ne parle pas dans cet article des problèmes de gestion de version liés à la modification d'un fichier. En effet, votre application peut être utilisée par plusieurs sociétés, utilisant des versions différentes. Les problèmes de gestion de version nécessiteraient un livre à eux seuls, et dépassent l'objectif de cet article qui est de vous expliquer comment utiliser la commande CHGPF pour modifier un fichier DB2/400.

La phase la plus délicate arrive ensuite, en effet, il faut modifier sur un AS/400 de production, un fichier contenant des milliers d'enregistrements (voire des dizaines ou des centaines de milliers), et utilisant au mieux une dizaine de fichiers logiques, au pire plusieurs dizaines. Si vous ne disposez que d'une "fenêtre" réduite pour effectuer la mise à jour sur un AS/400 de production, et si vous avez du mal à estimer le temps que prendra cette mise à jour, vous pouvez copier le fichier à mettre à jour (si sa taille, et la place disponible sur l'AS/400 le permettent) dans une bibliothèque de test et effectuer la mise à jour du fichier dans cette bibliothèque.

Comment effectuer la mise à niveau du fichier ? Elle se fait généralement en plusieurs étapes qui sont :

1. supprimer les fichiers logiques attachés au fichier physique à remplacer
2. renommer ou dupliquer le fichier physique à remplacer (pour ma part je préfère renommer pour économiser l'espace disque)
3. compiler le fichier physique si la modification est faite en environnement de développement, ou copier la nouvelle version du fichier en attente dans une bibliothèque tampon s'il s'agit d'une mise à jour d'environnement de production
4. copier les données de l'ancien fichier vers le nouveau par la commande CPYF en utilisant les paramètres *MAP *DROP. Le SGBD mettra en correspondance les colonnes de même nom et remplira les nouvelles colonnes à blanc ou à zéro selon leur type.
5. compiler les fichiers logiques si la modification est faite en environnement de développement, ou copier les nouvelles versions des fichiers logiques en attente dans une bibliothèque tampon s'il s'agit d'une mise à jour d'environnement de production

Comment automatiser toutes ces manipulations rébarbatives ?

- Un premier moyen consiste à utiliser mon utilitaire MAJFIC dont vous trouverez le descriptif (et les sources) sur ce site.

- Un autre moyen, qui est quand même l'objet de cet article, est d'utiliser la commande CHGPF comme ci-dessous :

```
CHGPF FILE(BIBFIC/FILEPF) SRCFILE(BIBSRC/QDDSSRC) SRCMBR(FILEPF)
```

Comment ça marche ? Cette commande très puissante extrait du fichier source QDDSSRC de la bibliothèque BIBSRC, le membre FILEPF (source DDS du fichier à mettre à jour) et s'appuie sur ce membre pour effectuer la mise à jour du fichier physique FILEPF de la bibliothèque BIBFIC.

Avantages : très peu de manipulations, la mise à niveau (remplacement du fichier physique, mise à niveau des

logiques et recopie des données) est effectuée par le SGBD lui-même. Les fichiers logiques ne sont pas modifiés, sauf si les nouvelles colonnes se trouvent dans leurs clés, et la mise à niveau se trouve accélérée.

Inconvénients :

- pour que ça marche, vous êtes obligé de transférer le fichier QDDSSRC sur l'AS/400 où doit se faire la mise à jour, mais ça c'est un inconvénient mineur.
- plus embêtant, la mise à niveau du fichier se base sur les DDS du fichier physique, mais dans ces DDS, aucune mention n'est faite des fichiers logiques modifiés ou supprimés, ou de nouveaux fichiers logiques qui n'existaient pas sur l'ancienne version du fichier. On va donc être obligé de modifier, de supprimer ou d'ajouter manuellement les logiques qui n'auront pas été pris en compte par le CHGPF. C'est pourquoi, malgré les avantages du CHGPF, je me sers encore très souvent de mon utilitaire MAJFIC pour la phase de mise en production d'un fichier modifié.

N.B. : vous trouverez sur la page ci-contre une technique vous permettant d'identifier les fichiers DB2/400 qui ne sont pas au même niveau dans un environnement de production par rapport à un environnement de référence.