Mise à jour de fichiers DB2/400

- Les articles -

Date de mise en ligne : mardi 10 août 2004

Date de parution : 7 novembre 2003

Description :

Automatisation de la mise à niveau d'un fichier DB2/400 sur un environnement de production.

Environnement iSeries

La mise à niveau d'un fichier DB2/400 sur un environnement (généralement il s'agit d'un environnement de production) est une opération délicate lorsqu'on doit la faire manuellement. Aussi un programme automatisant autant que possible la mise à niveau de fichier DB2/400 me semble être bienvenu. Le programme CL MAJFICCL s'acquitte honorablement de cette tâche délicate.

ATTENTION :

Le CL décrit ci-dessous ne contrôle pas que le fichier à mettre à niveau est en cours d'utilisation (si le fichier est verrouillé par un utilisateur, le CL se plantera de toute façon). A vous de le contrôler avant de lancer la mise à jour. Cela peut faire l'objet d'une amélioration pour une version ultérieure. De même, le CL ne se préoccupe pas des fichiers joints (j'aborderai ce problème plus loin dans la note).

Il faut noter que je fais ici un usage intensif de la commande SNDPGMMSG, ceci afin de connaître à tout moment l'état d'avancement de la mise à niveau. Ainsi, si le traitement de mise à niveau se plante à un moment donné, vous pourrez facilement savoir à quel moment et pourquoi il s'est planté.

A noter que si le fichier à mettre à niveau comporte plusieurs membres, le CL de mise à niveau le gère sans problème.

Voyons plus en détail le fonctionnement de MAJFICCL :

MAJFICCL reçoit 4 paramètres :

1. &BIBREF = bibliothèque d'origine (10 caractères maxi)
2. &BIBMAJ = bibliothèque de destination (10 caractères maxi)
3. &FIPHY = fichier physique à mettre à niveau (10 car. maxi ; ex. : CLIENT)
4. &FILOG = FICHIER(S) LOGIQUE(S) (10 car. maxi ; exemple : CLIENTL*)
5. &FILOG2 = FICHIER(S) LOGIQUE(S) (10 car. maxi ; exemple : CLIENTS*)

Déroulement des opérations :

1. Contrôle de l'existence des bibliothèques BIBMAJ et BIBREF, auquel cas le programme CL vous envoie un message et la mise à jour s'arrête,
2. Le programme contrôle que le fichier FIPHY à reprendre dans BIBREF existe bien,
3. Si tout va bien et si FILOG est renseigné, il passe à la suppression des fichiers logiques rattachés au fichier FIPHY de la bibliothèque BIBMAJ,
4. Opération importante : il renomme le fichier FIPHY de la bibliothèque BIBMAJ en plaçant le caractère "£" devant le nom du fichier. Ainsi si un problème devait survenir, les données pourront dans tous les cas être récupérées,
5. Copie du fichier FIPHY de la bibliothèque BIBREF vers la bibliothèque BIBMAJ,
6. Copie du fichier FILOG de la bibliothèque BIBREF vers la bibliothèque BIBMAJ. ATTENTION : si vous avez plusieurs fichiers logiques à mettre à jour comme par exemple les fichiers CLIENTL1, CLIENTL2 et CLIENTL3, il est tout à fait possible d'indiquer CLIENTL* en paramètre pour FILOG. Le CL s'en débrouillera très bien.
7. Copie du fichier FILOG2 de la bibliothèque BIBREF vers la bibliothèque BIBMAJ. ATTENTION : si vous avez plusieurs fichiers logiques à mettre à jour comme par exemple les fichiers CLIENTS1, CLIENTS2 et CLIENTS3, il est tout à fait possible d'indiquer CLIENTS* en paramètre pour FILOG2. Le CL s'en débrouillera très bien.
8. Récupération des données du fichier de sauvegarde (par exemple : £CLIENT) vers le nouveau fichier (par

exemple : CLIENT) via la commande CPYF (avec les paramètres *MAP et *DROP pour éviter un plantage d- aux différences de niveau entre £CLIENT et CLIENT),

9. Si tout s'est bien passé, suppression du fichier de sauvegarde (£CLIENT dans l'exemple),
10. Envoi à l'écran d'un message d'autosatisfaction, de la part du CL, comme quoi tout s'est bien passé.

Confort d'utilisation

Pour faciliter l'utilisation du CL MAJFICCL, j'ai créé une CMD (commande) appelée MAJFIC dont voici le source :

```
CMD        PROMPT(' MISE A NIVEAU DE FICHIER DB2 ')
PARM       KWD(BIBREF) TYPE(*CHAR) LEN(10) MIN(1) +
PROMPT('BIBLIOTHÈQUE D''ORIGINE. . .')
PARM       KWD(BIBMAJ) TYPE(*CHAR) LEN(10) MIN(1) +
PROMPT('BIBLIOTHÈQUE DE DESTINATION')
PARM       KWD(FIPHY) TYPE(*CHAR) LEN(10) MIN(1) +
PROMPT('FICHIER PHYSIQUE . . ')
PARM       KWD(FILOG) TYPE(*CHAR) LEN(10) +
CHOICE('CLIENTL1 , CLIENTL* , ETC...') +
PROMPT('FICHIER(S) LOGIQUE(S)')
PARM       KWD(FILOG2) TYPE(*CHAR) LEN(10) +
CHOICE('CLIENTS1 , CLIENTS* , ETC...') +
PROMPT('FICHIER(S) LOGIQUE(S)')
```

Source du CL MAJFICCL :

```
/*-------------------------------------------------------------------*/
 /*                                                                   */
 /*  MISE EN PLACE FICHIERS MODIFIES                                  */
 /*   PARAMETRES : &BIBREF = BIBLIOTHEQUE D'ORIGINE                   */
 /*                &BIBMAJ = BIBLIOTHEQUE DE DESTINATION              */
 /*                &FIPHY  = FICHIER PHYSIQUE (EX : CLIENT)           */
 /*                &FILOG  = FICHIER(S) LOGIQUE(S) ( EX : CLIENTL* )*/
 /*                &FILOG2 = FICHIER(S) LOGIQUE(S) ( EX : CLIENTS* )*/
 /*                                                                   */
 /*-------------------------------------------------------------------*/
 PGM        PARM(&BIBREF &BIBMAJ &FIPHY &FILOG &FILOG2)

 DCL        VAR(&BIBREF) TYPE(*CHAR) LEN(10)
 DCL        VAR(&BIBMAJ) TYPE(*CHAR) LEN(10)
 DCL        VAR(&FIPHY) TYPE(*CHAR) LEN(10)
 DCL        VAR(&FILOG) TYPE(*CHAR) LEN(10)
 DCL        VAR(&FILOG2) TYPE(*CHAR) LEN(10)
 DCL        VAR(&SVPHY) TYPE(*CHAR) LEN(10)
 DCL        VAR(&TAG) TYPE(*CHAR) LEN(1)
 DCL        VAR(&NBREC) TYPE(*DEC) LEN(10 0)

 CHGVAR     VAR(&SVPHY) VALUE('£' *CAT &FIPHY)

 SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
 MSGDTA('MISE À NIVEAU DU FICHIER ' *CAT +
 &FIPHY *CAT ' DE LA BIBLIOTHÈQUE ' *CAT +
 &BIBMAJ) TOPGMQ(*EXT) MSGTYPE(*STATUS)

 /*-------------------------------------------------------------------*/
```

```
/* - CONTROLE BIBLIOTHEQUE &BIBMAJ                          - */

IF         COND(&BIBMAJ = '          ') THEN(DO)
GOTO       CMDLBL(FIN)
ENDDO

CHKOBJ     OBJ(QSYS/&BIBMAJ) OBJTYPE(*LIB)
MONMSG     MSGID(CPF9801) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('NOM DE LA BIBLIOTHÈQUE À METTRE À +
JOUR INCORRECT ' *CAT &BIBMAJ) +
TOPGMQ(*PRV) MSGTYPE(*INFO)
GOTO FIN
ENDDO

/*-------------------------------------------------------------*/
/* - CONTROLE BIBLIOTHEQUE &BIBREF                          - */

IF         COND(&BIBREF = '          ') THEN(DO)
GOTO       CMDLBL(FIN)
ENDDO

CHKOBJ     OBJ(QSYS/&BIBREF) OBJTYPE(*LIB)
MONMSG     MSGID(CPF9801) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('NOM DE LA BIBLIOTHÈQUE DE +
RÉFÉRENCE INCORRECT ' *CAT &BIBREF) +
TOPGMQ(*PRV) MSGTYPE(*INFO)
GOTO FIN
ENDDO

MONMSG     MSGID(CPF2103)

/*-------------------------------------------------------------*/
/* - CONTROLE PHYSIQUE A REPRENDRE DANS BIBLIOTHEQUE &BIBREF   - */

CHKOBJ     OBJ(&BIBREF/&FIPHY) OBJTYPE(*FILE)
MONMSG     MSGID(CPF9801) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('MANQUE FICHIER ' *CAT &FIPHY *CAT +
' DANS BIBLIOTHÈQUE ' *CAT &BIBREF) +
TOPGMQ(*PRV) MSGTYPE(*INFO)
GOTO FIN
ENDDO

/*-------------------------------------------------------------*/
/* - SUPPRESSION DES FICHIERS LOGIQUES                      - */

IF         COND(&FILOG *NE ' ') THEN(DO)
DLTF       FILE(&BIBMAJ/&FILOG)
MONMSG     MSGID(CPF2125)
MONMSG     MSGID(CPF2105)
```

```
ENDDO

IF         COND(&FILOG2 *NE ' ') THEN(DO)
DLTF       FILE(&BIBMAJ/&FILOG2)
MONMSG     MSGID(CPF2125)
MONMSG     MSGID(CPF2105)
ENDDO

/*-------------------------------------------------------------*/
/* - RENAME DU FICHIER PHYSIQUE &FIPHY                       - */

CHGVAR     VAR(&TAG) VALUE('0')
RNMOBJ     OBJ(&BIBMAJ/&FIPHY) OBJTYPE(*FILE) +
NEWOBJ(&SVPHY)

MONMSG     MSGID(CPF2105) EXEC(DO)
CHGVAR     VAR(&TAG) VALUE('1')
ENDDO

MONMSG     MSGID(CPF3201) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) MSGDTA('IL +
EXISTE DÉJÀ UNE SAUVEGARDE DU FICHIER ' +
*CAT &FIPHY) TOPGMQ(*PRV) MSGTYPE(*INFO)
GOTO FIN
ENDDO

IF         COND(&TAG *EQ '0') THEN(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('FICHIER ' *CAT &FIPHY *CAT ' +
RENOMMÉ EN ' *CAT &SVPHY) TOPGMQ(*EXT) +
MSGTYPE(*STATUS)
ENDDO

/*-------------------------------------------------------------*/
/* - COPIE DES OBJETS FICHIERS (PHYSIQUE ET LOGIQUE)         - */

CRTDUPOBJ  OBJ(&FIPHY) FROMLIB(&BIBREF) OBJTYPE(*FILE) +
TOLIB(&BIBMAJ)
MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('ERREUR AU COURS DU CRTDUPOBJ DU +
FICHIER ' *CAT &FIPHY) TOPGMQ(*PRV) +
MSGTYPE(*INFO)
GOTO FIN
ENDDO
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('FICHIER PHYSIQUE ' *CAT &FIPHY +
*CAT ' DUPLIQUÉ') TOPGMQ(*EXT) +
MSGTYPE(*STATUS)

IF         COND(&FILOG *NE ' ') THEN(DO)
CRTDUPOBJ  OBJ(&FILOG) FROMLIB(&BIBREF) OBJTYPE(*FILE) +
```

```
TOLIB(&BIBMAJ)
MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('ERREUR AU COURS DU CRTDUPOBJ DU +
FICHIER ' *CAT &FILOG) TOPGMQ(*PRV) +
MSGTYPE(*INFO)
GOTO FIN
ENDDO

SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('FICHIER LOGIQUE ' *CAT &FILOG +
*CAT ' DUPLIQUÉ') TOPGMQ(*EXT) +
MSGTYPE(*STATUS)
ENDDO

IF         COND(&FILOG2 *NE ' ') THEN(DO)
CRTDUPOBJ  OBJ(&FILOG2) FROMLIB(&BIBREF) OBJTYPE(*FILE) +
TOLIB(&BIBMAJ)
MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('ERREUR AU COURS DU CRTDUPOBJ DU +
FICHIER ' *CAT &FILOG2) TOPGMQ(*PRV) +
MSGTYPE(*INFO)
GOTO FIN
ENDDO

SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('FICHIER LOGIQUE ' *CAT &FILOG2 +
*CAT ' DUPLIQUÉ') TOPGMQ(*EXT) +
MSGTYPE(*STATUS)
ENDDO
/*-------------------------------------------------------------*/
/* - TEST SI FICHIER CONTIENT DONNEES A RECUPERER             - */

IF         COND(&TAG *EQ '1') THEN(GOTO CMDLBL(SUITE))
RTVMBRD    FILE(&BIBMAJ/&SVPHY) NBRCURRCD(&NBREC)
IF         COND(&NBREC *EQ 0) THEN(GOTO CMDLBL(SUITE))

/*-------------------------------------------------------------*/
/* - RECUPERATION DES DONNEES                                 - */

CPYF       FROMFILE(&BIBMAJ/&SVPHY) +
TOFILE(&BIBMAJ/&FIPHY) FROMMBR(*ALL) +
TOMBR(*FROMMBR) MBROPT(*REPLACE) +
CRTFILE(*NO) FMTOPT(*MAP *DROP)

MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('ATTENTION : PROBLÈME DANS LA +
RÉCUPÉRATION DES DONNÉES DU FICHIER ' +
*CAT &FIPHY) TOPGMQ(*PRV) MSGTYPE(*INFO)
GOTO FIN
```

```
 ENDDO

 SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
 MSGDTA('DONNÉES DU FICHIER ' *CAT &FIPHY +
 *CAT ' RÉCUPÉRÉES') TOPGMQ(*EXT) +
 MSGTYPE(*STATUS)

 /*-------------------------------------------------------------*/
 /* - FIN DE TRAITEMENT                                         */

 SUITE:      DLTF       FILE(&BIBMAJ/&SVPHY)
 MONMSG     MSGID(CPF2105)

 SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
 MSGDTA('TRAVAIL DE MISE À NIVEAU SUR +
 FICHIER ' *CAT &FIPHY *CAT ' TERMINÉ +
 NORMALEMENT') TOPGMQ(*PRV) MSGTYPE(*INFO)
 FIN:        ENDPGM
```

Quelques améliorations :

Tout ça c'est très bien dans le cadre d'une mise à jour de fichier ponctuelle, mais pas satisfaisant lorsque vous avez tout un module à livrer avec une demi-douzaine de fichiers à mettre à niveau et autant de nouveaux fichiers à mettre en place (avec les risques d'erreur et d'oubli qui vont avec). C'est pourquoi, j'ai employé à plusieurs reprises une procédure de mise à niveau de fichier simple (car elle ne demandait pas un gros développement) et pratique (car utilisant une version légèrement modifiée du CL MAJFICCL).

A partir du CL MAJFICCL, nous avons élaboré un second CL (le MAJFIC2CL) qui a pour particularité d'avoir un paramètre supplémentaire : &ERR = Code erreur ("00" = Tout va bien ; "01" = GAME OVER). Je ne vous donne pas ici le source du CL MAJFIC2CL, car les différences avec le CL MAJFICCL sont mineures.

Comme vous l'avez compris, si &ERR est retourné avec "01", c'est que quelque chose s'est mal passé. A partir de là, il ne nous restait plus qu'à créer un CL de mise à niveau des fichiers utilisant MAJFIC2CL. Par convention, nous avons décidé d'appeler tous nos CL de mise à niveau "PTF" avec un numéro d'ordre sur 6 positions de façon à pouvoir les archiver. Ainsi un même CL peut servir à mettre à niveau plusieurs environnements clients, d'où un gain de temps appréciable et une diminution des risques d'erreur ou d'oubli durant la mise à jour. Pour illustrer mon propos, voici un exemple de CL, le PTF00001, qui fait les opérations suivantes (on suppose que les nouveaux fichiers se trouvent dans la bibliothèque BIBREFMAJ, comme indiqué dans le CL) :

1. Mise à jour du fichier physique F130 (qui a des logiques F130L*) et du fichier physique F1301 (qui n'en a pas)
2. Mise à jour d'un fichier logique dans l'environnement client (voir le fichier F101L3) sur un physique qui existe déjà (la mise à jour d'un logique seul pourrait faire l'objet d'un CL spécifique)
3. Mise en place d'un fichier joint (le F211J1) qui n'existait pas dans l'environnement client.

Source du CL PTF000001 :

```
PGM        PARM(&BIBCLI)
 DCL        VAR(&BIBCLI) TYPE(*CHAR) LEN(10)
 DCL        VAR(&BIBREF) TYPE(*CHAR) LEN(10) +
 VALUE('BIBREFMAJ ')
 DCL        VAR(&FIPHY)  TYPE(*CHAR) LEN(10)
 DCL        VAR(&FILOG)  TYPE(*CHAR) LEN(10)
 DCL        VAR(&FILOG2) TYPE(*CHAR) LEN(10)
 DCL        VAR(&ERR)    TYPE(*CHAR) LEN(2)
```

```
/*-------------------------------------------------------------*/
/* - MISE A JOUR DES FICHIERS                                  */
/*-------------------------------------------------------------*/
CHGVAR     VAR(&ERR)    VALUE('00')
CHGVAR     VAR(&FIPHY)  VALUE('CLIENT    ')
CHGVAR     VAR(&FILOG)  VALUE('CLIENTL*  ')
CHGVAR     VAR(&FILOG2) VALUE('CLIENTS*  ')
CALL       PGM(MAJFIC2CL) PARM(&BIBREF &BIBCLI &FIPHY +
&FILOG &ERR)
IF         COND(&ERR *NE '00') THEN(GOTO CMDLBL(FIN))
/*-------------------------------------------------------------*/
CHGVAR     VAR(&ERR) VALUE('00')
CHGVAR     VAR(&FIPHY) VALUE('TARCLI    ')
CHGVAR     VAR(&FILOG) VALUE('          ')
CHGVAR     VAR(&FILOG) VALUE('          ')
CALL       PGM(MAJFIC2CL) PARM(&BIBREF &BIBCLI &FIPHY +
&FILOG &ERR)
IF         COND(&ERR *NE '00') THEN(GOTO CMDLBL(FIN))
/*-------------------------------------------------------------*/
CHGVAR     VAR(&FILOG) VALUE('F101L3    ')
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('Ajout du logique ' *CAT &FILOG) +
TOPGMQ(*EXT) MSGTYPE(*STATUS)
DLTF       FILE(&BIBCLI/&FILOG)
MONMSG     MSGID(CPF2105)
CRTDUPOBJ  OBJ(&FILOG) FROMLIB(&BIBREF) OBJTYPE(*FILE) +
TOLIB(&BIBCLI)
MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('Erreur au cours du CRTDUPOBJ du +
fichier ' *CAT &FILOG) TOPGMQ(*PRV) +
MSGTYPE(*INFO)
GOTO       CMDLBL(FIN)
ENDDO
/*-------------------------------------------------------------*/
CRTDUPOBJ  OBJ(F211J1) FROMLIB(&BIBREF) OBJTYPE(*FILE) +
TOLIB(&BIBCLI)
MONMSG     MSGID(CPF0000) EXEC(DO)
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('Erreur au cours du CRTDUPOBJ du +
fichier F211J1') TOPGMQ(*PRV) +
MSGTYPE(*INFO)
GOTO FIN
ENDDO
/*-------------------------------------------------------------*/
/* - FIN DU PROGRAMME              - */
/*-------------------------------------------------------------*/
SNDPGMMSG  MSGID(CPF9897) MSGF(QSYS/QCPFMSG) +
MSGDTA('Mise à jour de la bibliothèque ' +
*CAT &BIBCLI *CAT ' terminée +
normalement') TOPGMQ(*PRV) MSGTYPE(*INFO)
```

```
FIN:        ENDPGM
```

Améliorations possibles :

Ce composant de mise à niveau de fichier est très bien si vos noms de fichiers physiques et logiques sont bien normalisés. Exemple : fichier physique CLIENT, fichiers logiques CLIENTL*, fichiers logiques sélectifs CLIENTS*. Mais dans le cas d'un fichier physique ayant un grand nombre de fichiers logiques dépendants avec des noms exotiques, ça ne marche pas (à moins de prévoir un grand nombre de paramètres FILOG, mais là ça devient "usine à gaz"). Une solution pourrait consister à extraire dans un fichier, grâce à la commande DSPDBR, la liste des fichiers dépendants du fichier physique à mettre à niveau, et à se baser sur ce fichier de dépendances pour mettre à niveau les fichiers logiques. En fait, c'est un peu plus compliqué que ça : il faudrait dans un premier temps extraire la liste des fichiers dépendants du fichier destination pour les supprimer, puis extraire la liste des fichiers dépendants du fichier d'origine pour les copier dans la bibliothèque du fichier destination. Voilà un petit développement intéressant, qui s'y colle ?

N.B. : vous trouverez en cliquant ici 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.