Contrôler la validité des Jobd (via SQL et l'API QWDRJOBD)

- Les articles -

Date de mise en ligne : lundi 1er septembre 2014

Description :

En tant qu'administrateur système, vous êtes peut être confronté au problème de la gestion des Jobd. Comme chacun sait, ces objets IBMi permettent de stocker des listes de bibliothèques. C'est très pratique, mais cela peut vite devenir l'enfer, car il n'est pas du tout évident de contrôler que les Jobd pointent toutes sur des bibliothèques valides. Je vous propose donc un petit outil de contrôle composé d'un peu de code SQL, RPG et CL.

Environnement iSeries

Les objets IBMi de type Jobd (Job Description) sont très pratiques, et donc très utilisés.

Rattachés à des profils utilisateurs et/ou à des travaux IBMi (batchs ou interactifs), ils permettent de définir la liste des bibliothèques que chaque travail est habilité à utiliser. D'un point de vue SQL, le fait d'utiliser la convention d'appellation système (*SYS) permet de s'appuyer sur cette notion de liste de bibliothèque. On peut dès lors s'affranchir de la notion de bibliothèque dans les programmes de type SQLRPGLE, ainsi que dans les procédures stockées SQL. Dans ce cas de figure, si un objet DB2 sur lequel on exécute une requête SQL n'a pas de bibliothèque définie, DB2 recherchera dans la liste des bibliothèques le premier objet portant le nom de l'objet considéré.

Ce mécanisme est très pratique, mais les Jobd ne sont pas des objets SQL, on ne peut donc pas exécuter de requête SQL permettant - par exemple - de retrouver toutes les Jobd utilisant une bibliothèque particulière. Si on souhaite renommer ou supprimer une bibliothèque, et si l'on a beaucoup de Jobd à contrôler, on court le risque de rendre certaines Jobd inopérantes à cause d'une bibliothèque manquante, et de voir certains traitements se planter lors de leur exécution.

En s'appuyant sur une API IBM (que l'on invoquera au travers d'un programme RPG), et un peu de CL et de SQL, on peut pallier ce manque, et développer un outil de contrôle fiable et performant.

On a besoin dans un premier temps d'un programme permettant d'extraire la liste des bibliothèques d'une jobd. Ce programme sera écrit en RPG en utilisant l'API IBM QWDRJOBD. Le source du programme RPG est donné un peu plus bas dans l'article.

Ce programme sera appelé au travers d'un programme CL, lui même associé à une UDF externe RTVJOBDSQF. Cette fonction sera appelée par l'UDTF RTVJOBDLIBL, cette dernière étant en mesure de renvoyer la liste des bibliothèques de la Jobd sous forme de liste SQL, en utilisant une technique récursive (dont j'ai parlé dans un précédent dossier XDocs).

Voici le source du CL MABIBPGM/QCLSRC MBR(RTVJOBDSQC) qui fait appel au programme RPGLE RTVJOBDPGM pour l'extraction de la liste des bibliothèques :

```
PGM        PARM(&JOBDNAME &JOBDLIB &LIBL)

 DCL        VAR(&JOBDNAME) TYPE(*CHAR) LEN(10)
 DCL        VAR(&JOBDLIB) TYPE(*CHAR) LEN(10)
 DCL        VAR(&VAR) TYPE(*CHAR) LEN(2750)
 DCL        VAR(&LIBL) TYPE(*CHAR) LEN(2750)
 DCL        VAR(&LIBLCNT) TYPE(*INT) LEN(2)
 DCL        VAR(&JOBDL) TYPE(*CHAR) LEN(20)
 CHGVAR     VAR(&JOBDL) VALUE(&JOBDNAME *CAT &JOBDLIB)

 CALL       PGM(RTVJOBDPGM) PARM(&JOBDL &LIBL +
 &LIBLCNT)

 ENDPGM
```

Là c'était facile, voici maintenant le source de la fonction externe encapsulant le CL RTVJOBDSQC pour appel du programme RPGLE RTVJOBDPGM (ce dernier faisant appel à l'API QWDRJOBD) :

```
CREATE OR REPLACE FUNCTION MABIBPGM/RTVJOBDSQF (
 JOBDNAME CHAR(10) ,
 JOBDLIB CHAR(10) )
 RETURNS CHAR(2750)
 LANGUAGE CL
 SPECIFIC MABIBPGM/RTVJOBDSQF
 NOT DETERMINISTIC
 MODIFIES SQL DATA
 CALLED ON NULL INPUT
 EXTERNAL NAME 'MABIBPGM/RTVJOBDSQC'
 PARAMETER STYLE SQL ;

 LABEL ON SPECIFIC FUNCTION MABIBPGM/RTVJOBDSQF
 IS 'Extraction liste des bibl. d''une Jobd' ;
```

Voici maintenant le source de la fonction SQL (de type UDTF) permettant d'extraire la liste des bibliothèques d'une Jobd sous forme d'une "table function".

```
CREATE OR REPLACE FUNCTION MABIBPGM/RTVJOBDLIBL (
 JOBDNAME VARCHAR(10) ,
 JOBDLIB VARCHAR(10) )
 RETURNS TABLE (
 JOBDNOUT CHAR(10) ,
 JOBDLOUT CHAR(10) ,
 LIBNAME CHAR(10)
 )
 LANGUAGE SQL
 SPECIFIC MABIBPGM/RTVJBDLIBL
 NOT DETERMINISTIC
 MODIFIES SQL DATA
 CALLED ON NULL INPUT
 SET OPTION  ALWBLK = *ALLREAD ,
 ALWCPYDTA = *OPTIMIZE ,
 COMMIT = *NONE ,
 DECRESULT = (31, 31, 00) ,
 DFTRDBCOL = *NONE ,
 DYNDFTCOL = *NO ,
 DYNUSRPRF = *USER ,
 SRTSEQ = *HEX
 BEGIN

 DECLARE LIBL CHAR ( 2750 ) DEFAULT '' ;
 DECLARE JOBDNTMP CHAR ( 10 ) ;
 DECLARE JOBDLTMP CHAR ( 10 ) ;

 -- Passage des paramètre de type VARCHAR en CHAR pour faciliter l'utilisation
 --  de la fonction RTVJOBDSQF qui ne tolère pas les VARCHAR
 SET JOBDNTMP = JOBDNAME ;
 SET JOBDLTMP = JOBDLIB ;

 -- Table temporaire pour extraire la liste des bibliothèques
 DECLARE GLOBAL TEMPORARY TABLE TMPLIBLIST
 ( LIBNAME CHAR ( 10 ) )
```

```
WITH REPLACE ;

-- Récupération de la liste des bibliothèques de la jobd au format CHAR ( 2750 )
SET LIBL = ( SELECT RTVJOBDSQF ( JOBDNTMP , JOBDLTMP ) FROM SYSIBM / SYSDUMMY1 ) ;

-- Utilisation de la récursivité pour extraire la liste des bibliothèques
INSERT INTO SESSION / TMPLIBLIST ( LIBNAME )
WITH GEN_IDS ( NX ) AS (
SELECT 1 AS N1 FROM SYSIBM / SYSDUMMY1
UNION ALL
SELECT NX + 11 AS N2 FROM GEN_IDS WHERE NX < 2750
)
SELECT
SUBSTR ( LIBL , NX , 10 ) AS LIBLIST
FROM GEN_IDS ;

RETURN
WITH TMP AS (
SELECT JOBDNAME , JOBDLIB , LIBNAME FROM SESSION / TMPLIBLIST
)
SELECT JOBDNAME , JOBDLIB , LIBNAME FROM TMP ;
END ;

LABEL ON SPECIFIC FUNCTION MABIBPGM/RTVJBDLIBL
IS 'Extraction liste des bibl. d''une Jobd' ;
```

On passe maintenant au plat de résistance, avec le source du programme RPG. Ce programme avait été écrit par Robert Cozzi en 2006, je l'ai partiellement réécrit en RPG Free :

```
H DFTACTGRP(*NO) OPTION(*SRCSTMT : *NODEBUGIO)
H COPYRIGHT('(c) 2006 - Robert Cozzi, Jr. - All rights reserved.')

**************************************************************
**  RTVJOBD - Retrieve Job Description Command Proc Pgm.
**            This program returns the library list
**            of the specified Job Description ("jobd").
**            In addition, the number of library names
**            in the jobd's library lis is also returned.
**            See the associated RTVJOBD CMD source for use
**            in CL.  TIP: The return variables in your CL
**            program should be defined as follows:
**               DCL &LIBL    TYPE(*CHAR) LEN(2750)
**               DCL &LIBLCNT TYPE(*INT)  LEN(2)
**  Partially rewritten in RPG Free by Gregory Jarrige (2014-06-04)

D RtvJobD         PR
D  szJobD                       20A   Const
D  rtnLIBL                    2750A
D  rtnLIBLCount                  5I 0

D RtvJobD         PI
D  szJobD                       20A   Const
D  rtnLIBL                    2750A
```

```
D  rtnLIBLCount                 5I 0

/COPY QSYSINC/QRPGLESRC,QWDRJOBD
/COPY QSYSINC/QRPGLESRC,QUSEC

**  Retrieve Job Description
D*QWDRJOBD        PR                  ExtPgm('QWDRJOBD')
D RtvJobDAPI      PR                  ExtPgm('QWDRJOBD')
D  szRtnBuffer               65535A   OPTIONS(*VARSIZE)
D  nRtnBufLen                   10I 0 Const
D**  Specify 'JOBD0100'
D  apiFormat                     8A   Const
D  JobD                         20A   Const
D  api_error                          LikeDS(QUSEC)

D JobD            DS                  LikeDS(QWDD0100)
D                                     Based(pJobD)

D JobDInfo        DS                  LikeDS(QWDD0100)

D LibList         S             11A   Based(pLIBL) DIM(250)
D LibL            S           2750A   Based(pLIBL)
D APIErrDS        DS                  LikeDS(QUSEC)
/free
*INLR = *ON ;

//  Sadly, with this API, we need to call it twice when
//  the LIBL is needed.
//  First call: Get the length of the data to be returned.
APIErrDS = *ALLX'00' ;
APIErrDS.QUSBPRV = %size(APIErrDS) ;
JobDInfo = *ALLX'00' ;
callp RtvJobDAPI(JobDInfo : %size(JobDInfo):
'JOBD0100': szJOBD : APIErrDS) ;

if APIErrDS.QUSBAVL = 0 ;
pJobD = %Alloc(JobDInfo.QWDBAVL) ;
JOBD = *ALLX'00' ;
//  Second call: Get the library list.
callp RtvJobDAPI(JOBD : JobDInfo.QWDBAVL :
'JOBD0100': szJOBD : APIErrDS) ;
if %Parms >= 3 ;
rtnLIBLCount = JobD.QWDNLILL ;
endif ;
if %Parms >= 2 ;
pLibl = pJobD + JobD.QWDOILL;
rtnLibl = %subst(LIBL:1:JobD.QWDNLILL*%size(LibList));
endif ;
deAlloc pJobD ;
endif ;
/end-free
```

Ce n'est pas tout à fait terminé (vous ne pensiez pas vous en tirer comme ça ? ;)

Il nous reste à écrire une dernière UDTF, que nous appellerons CHKALLJOBD. Cette UDTF effectuera un DSPJOBD *ALL vers une table temporaire, ce qui nous permettra ensuite de lire le contenu de cette table temporaire, et pour chaque occurrence de la table, d'appeler notre fonction RTVJOBDLIBL permettant d'extraire la liste des bibliothèques déclarée au sein de la Jobd. Pour chacune des bibliothèques appartenant à la liste obtenue, nous ferons appel à une petite fonction SQL que j'ai appelée CHKOBJSQL et dont j'ai présenté le fonctionnement dans un précédent dossier. Je rappelle que cette petite fonction a pour but de vérifier si l'objet qu'on lui passe en paramètre existe bien sur le système (pour ce faire elle encapsule un CL qui fait un bête CHKOBJ). Si la fonction CHKOBJSQL indique que la jobd n'existe pas, alors elle fera partie des données renvoyées dans le result set produit par l'UDTF CHKALLJOBD.

L'utilisation de l'UDTF CHKALLJOBD se fera au moyen de la requête suivante :

```
select * from table ( MABIBPGM.CHKALLJOBD  () ) myudtf ;
```

Voici le code source de la fonction CHKALLJOBD, fonction destinée à contrôler la liste des bibliothèques pour l'ensemble des Jobd d'un système :

```
CREATE OR REPLACE FUNCTION MABIBPGM/CHKALLJOBD ( )
RETURNS TABLE (
JOBDNOUT CHAR(10) ,
JOBDLOUT CHAR(10) ,
LIBNAME CHAR(10) )
LANGUAGE SQL
SPECIFIC MABIBPGM/CHKALLJOBD
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
SET OPTION  ALWBLK = *ALLREAD ,
ALWCPYDTA = *OPTIMIZE ,
COMMIT = *NONE ,
DECRESULT = (31, 31, 00) ,
DFTRDBCOL = *NONE ,
DYNDFTCOL = *NO ,
DYNUSRPRF = *USER ,
SRTSEQ = *HEX
BEGIN
DECLARE SQLSTATE CHAR ( 5 ) DEFAULT '00000' ;
DECLARE VSQL VARCHAR ( 256 ) ;
DECLARE VCMD VARCHAR ( 256 ) ;
DECLARE VBIBL CHAR ( 10 ) ;
DECLARE VOBJET CHAR ( 10 ) ;

DECLARE CUR1 CURSOR FOR V_DYNSTM ;
DECLARE GLOBAL TEMPORARY TABLE TMPLIBLIS3 (
JOBDN CHAR ( 10 ) ,
JOBDL CHAR ( 10 ) ,
LIBNAME CHAR ( 10 ) ,
TROUVE CHAR ( 1 )
) WITH REPLACE ;

SET VCMD = 'DSPOBJD OBJ(*ALL/*ALL) OBJTYPE(*JOBD) OUTPUT(*OUTFILE) OUTFILE(QTEMP/TMPJOBDLST) OUTMBR(*FIRST
```

```
*REPLACE)' ;
 CALL QCMDEXC ( VCMD , LENGTH ( VCMD ) ) ;

 SET VSQL = 'SELECT ODLBNM as bibl, ODOBNM as objet FROM QTEMP.TMPJOBDLST' ;
 PREPARE V_DYNSTM FROM VSQL ;

 OPEN CUR1 ;
 FETCH CUR1 INTO VBIBL , VOBJET ;

 WHILE ( SQLSTATE = '00000' ) DO

 INSERT INTO SESSION / TMPLIBLIS3 ( JOBDN , JOBDL , LIBNAME , TROUVE )
 WITH TMPLIST AS (
 SELECT * FROM TABLE ( RTVJOBDLIBL ( VOBJET , VBIBL ) ) MYUDTF
 )
 SELECT JOBDNOUT , JOBDLOUT , LIBNAME ,
 CHKOBJSQL ( LIBNAME , '*LIBL' , '*LIB' ) AS TROUVE
 FROM TMPLIST WHERE TRIM ( LIBNAME ) <> '' AND SUBSTR ( LIBNAME , 1 , 1 ) <> '*' ;

 FETCH CUR1 INTO VBIBL , VOBJET ;

 END WHILE ;

 CLOSE CUR1 ;

 DELETE FROM SESSION / TMPLIBLIS3 WHERE TROUVE = '1' ;

 RETURN
 WITH TMP AS (
 SELECT JOBDN , JOBDL , LIBNAME FROM SESSION / TMPLIBLIS3
 )
 SELECT JOBDN , JOBDL , LIBNAME FROM TMP ;
 END
 ;

 LABEL ON SPECIFIC FUNCTION MABIBPGM/CHKALLJOBD
 IS 'Contrôle liste de bibl. / toutes les Jobd' ;
```

Et voilà, j'espère que cet outil pourra être utile au plus grand nombre. Bonnes compilations !