Récupérer la liste des travaux verrouillant une table DB2

- Les articles -

Date de mise en ligne : lundi 29 avril 2013

Description :

Comment récupérer la liste des travaux verrouillant une table DB2, et rendre cette liste accessible à des applications écrites en PHP, Java ou autre...

Environnement iSeries

Quand on souhaite savoir si une table DB2 pour IBM i est verrouillée, et surtout par quels travaux elle est verrouillée, la technique la plus couramment utilisée consiste à passer en mode "ligne de commande" et à utiliser la commande WRKOBJLCK. Cette commande est très pratique mais elle souffre de deux défauts :

- on est obligé de lui transmettre le nom court de la table DB2 à contrôler, ce qui devient vite agaçant quand on doit travailler avec des bases DB2 qui utilisent les noms longs
- elle ne permet pas d'exporter la liste des verrouillages dans une table

On va voir qu'il est relativement facile de contourner ces deux difficultés.

En ce qui concerne le problème du nom court tout d'abord, il est facile de récupérer ce nom court à partir de son nom long, au moyen d'une requête SQL du type :

```
SELECT SYSTEM_TABLE_NAME, SYSTEM_TABLE_SCHEMA
 FROM QSYS2/SYSTABLES
 WHERE table_name = ? and table_schema = ?
```

En ce qui concerne la récupération des verrouillages dans une table, on peut le faire au moyen d'un programme RPG faisant appel à l'API QWCLOBJL.

Le programme RPG va exploiter les données renvoyées par l'API QWCLOBJL, et va stocker ces données dans une table temporaire que j'ai appelée OBJL0100. On pourra exploiter le contenu de cette table temporaire après l'appel du programme RPG, dès lors que cette opération s'effectue dans le même travail que l'appel du programme RPG. Mais on peut aussi demander au programme RPG de générer un "result set", ce qui facilite la récupération des informations au niveau du programme appelant, qu'il s'agisse d'un script PHP ou d'un programme Java.

Pour écrire le programme RPG, je suis parti d'un exemple très complet proposé par l'expert IBM i Scott Klement, sur la page suivante :

http://archive.midrange.com/rpg400-l/200202/msg00344.html

J'ai retravaillé l'exemple de Scott Klement en y ajoutant les étapes de création et d'alimentation de la table temporaire. J'en ai profité aussi pour réécrire la majeure partie du code en RPG free (format libre). De plus, Scott Klement proposait dans son exemple d'afficher directement sur un écran de type 5250 les informations renvoyées par l'API, j'ai complètement supprimé cette partie "affichage".

Une petite astuce à noter : dans son exemple, Scott Klement avait placé les directives suivantes en début de programme : DFTACTGRP(*NO) ACTGRP(*NEW) Je me suis rendu compte à l'usage que ces directives m'empêchaient de récupérer mon result set, plus quelques autres effets de bord sur lesquels je ne m'étendrai car c'est un peu hors sujet (si vous êtes curieux, je vous laisse le soin de les découvrir par vous même). J'ai donc retiré ces directives.

Je vous livre ci-dessous le code source du programme RPG, et juste après, vous trouverez le code source de la procédure stockée DB2 de type externe, qui me permet d'encapsuler le programme RPG et de le rendre facilement accessible aux applications PHP, Java, ou autre...

Code source du programme RPG :

```
*****************************************************************
* Exemple d'utilisation de l'API QWCLOBJL (équivalent de WRKOBJLCK)
* Adaptation d'un exemple fourni par Scott Klement sur :
*  http://archive.midrange.com/rpg400-l/200202/msg00344.html
* Adapté par Gregory Jarrige le 22/04/2013
* Modifications par rapport à la version de Scott Klement :
* - réécriture du code en RPG Free
* - génération de la liste des "locks" dans une table temporaire DB2
* - renvoi optionnel de la liste des "locks" en result set
* - attention : ne surtout pas utiliser les directives de compilation
*   proposées par Scott Klement (DFTACTGRP(*NO) ACTGRP(*NEW)), car
*   elles ont pour effet d'empêcher le renvoi du "result set" vers
*   la procédure stockée DB2 qui encapsule le programme RPG.
*****************************************************************
H usrprf(*owner) datfmt(*iso)
D CrtUsrSpc       PR                  ExtPgm('QUSCRTUS')
D   UsrSpc                      20A   CONST
D   ExtAttr                     10A   CONST
D   InitSize                    10I 0 CONST
D   InitVal                      1A   CONST
D   PublicAuth                  10A   CONST
D   Text                        50A   CONST
D   Replace                     10A   CONST
D   ErrorCode                32766A   options(*varsize)

D RtvPtrUS        PR                  ExtPgm('QUSPTRUS')
D   UsrSpc                      20A   CONST
D   Pointer                       *

D LstObjLck       PR                  ExtPgm('QWCLOBJL')
D   UsrSpc                      20A   const
D   Format                       8A   const
D   Object                      20A   const
D   ObjType                     10A   const
D   Member                      10A   const
D   ErrorCode                32766A   options(*varsize)

D*************************************************
D* API error code data structure
D*************************************************
D dsEC            DS
D*                                    Bytes Provided (size of struct)
D  dsECBytesP             1      4I 0 INZ(256)
D*                                    Bytes Available (returned by API)
D  dsECBytesA             5      8I 0 INZ(0)
D*                                    Msg ID of Error Msg Returned
D  dsECMsgID              9     15
D*                                    Reserved
D  dsECReserv            16     16
D*                                    Msg Data of Error Msg Returned
D  dsECMsgDta            17    256
```

```
D*****************************************************
D* List API generic header data structure
D*****************************************************
D dsLH            DS                  BASED(p_UsrSpc)
D*                                    Filler
D   dsLHFill1                  103A
D*                                    Status (I=Incomplete,C=Complete
D*                                           F=Partially Complete)
D   dsLHStatus                  1A
D*                                    Filler
D   dsLHFill2                  12A
D*                                    Header Offset
D   dsLHHdrOff                 10I 0
D*                                    Header Size
D   dsLHHdrSiz                 10I 0
D*                                    List Offset
D   dsLHLstOff                 10I 0
D*                                    List Size
D   dsLHLstSiz                 10I 0
D*                                    Count of Entries in List
D   dsLHEntCnt                 10I 0
D*                                    Size of a single entry
D   dsLHEntSiz                 10I 0

D*****************************************************
D*  List Object Locks API format OBJL0100
D*****************************************************
D* http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Fapis%2Fqwclobjl.htm
D dsOL            DS                 based(p_Entry)
D*                                    Job Name
D  dsOL_JobName                 10A
D*                                    Job User Name
D  dsOL_UserName                10A
D*                                    Job Number
D  dsOL_JobNbr                   6A
D*                                    Lock State
D  dsOL_LckState                10A
D*                                    Lock Status
D  dsOL_LckSts                  10i 0
D*                                    Lock Type
D  dsOL_LckType                 10i 0
D*                                    Member (or *BLANK)
D  dsOL_Member                  10A
D*                                    1=Shared File, 0=Not Shared
D  dsOL_Share                    1A
D*                                    Lock Scope
D  dsOL_LckScope                 1A
D*                                    Thread identifier
D  dsOL_ThreadID                 8A

D p_UsrSpc        S                *
D p_Entry         S                *
```

```
D Msg             S             50A
D MsgLockStatus   S             10A
D MsgLockType     S             11A
D MsgShare        S              4A
D MsgScope        S             10A
D Sql1            S            450A
D Sql2            S            100A
D Sql3            S            450A
D x               S             10I 0

C     *entry        plist
C                   parm                    ObjName          10
C                   parm                    ObjLib           10
C                   parm                    ObjType          10
C                   parm                    Member           10
C                   parm                    Resultset         3

/free
If ( %parms < 5 ) ;
*inlr = *on ;
Endif ;

// ****************************************
// Create a user space to store output of
// the list object locks API
// ****************************************
callp     CrtUsrSpc('OBJLOCKS  QTEMP': 'USRSPC':
1: x'00': '*ALL': 'Output of List ' +
'Object Locks API': '*YES': dsEC) ;
If ( dsECBytesA > 0 ) ;
*inlr = *on ;
Endif ;

// ****************************************
// Dump the Object Locks to the user space
// ****************************************
callp     LstObjLck('OBJLOCKS  QTEMP': 'OBJL0100':
ObjName+ObjLib: ObjType: Member: dsEC) ;

If ( dsECBytesA > 0 ) ;
*inlr = *on ;
Endif ;

// ****************************************
// Get a pointer to the user space
// ****************************************

callp     RtvPtrUS('OBJLOCKS  QTEMP': p_UsrSpc) ;

//*********************************************
// Création d'une table DB2 temporaire destinée
// à stocker les entrées de la liste renvoyée
```

```
// par l'API
//*******************************************
sql1 = 'declare global temporary table OBJL0100 ( ' +
'Job_name        CHAR(10) , ' +
'Job_user_name   CHAR(10) , ' +
'Job_number      CHAR(6)  , ' +
'Lock_state      CHAR(10) , ' +
'Lock_status     CHAR(10) , ' +
'Lock_type       CHAR(11) , ' +
'Member_name     CHAR(10) , ' +
'Share           CHAR(4)  , ' +
'Lock_scope      CHAR(10) , ' +
'Thread_id       CHAR(8) ' +
') with replace ' ;

EXEC SQL EXECUTE IMMEDIATE :sql1 ;

sql2 = 'INSERT INTO QTEMP/OBJL0100 ' +
'VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)' ;

EXEC SQL PREPARE cur1 FROM :sql2 ;

for x = 0 to (dsLHEntCnt-1) ;
p_Entry = p_UsrSpc + (dsLHLstOff + (dsLHEntSiz*x));

MsgLockStatus = '' ;
select ;
when      dsOL_LckSts = 1 ;
MsgLockStatus = 'HELD' ;
when      dsOL_LckSts = 2 ;
MsgLockStatus = 'WAIT' ;
when      dsOL_LckSts = 2 ;
MsgLockStatus = 'REQ' ;
endsl;

MsgLockType = '' ;
select ;
when      dsOL_LckType = 1 ;
MsgLockType = 'OBJECT';
when      dsOL_LckType = 2 ;
MsgLockType = 'MBR CTL BLK';
when      dsOL_LckType = 3 ;
MsgLockType = 'MBR ACC PTH';
when      dsOL_LckType = 3 ;
MsgLockType = 'MBR DATA' ;
endsl;

If ( dsOL_Share = '1' ) ;
MsgShare = 'YES' ;
Else ;
MsgShare = 'NO' ;
Endif ;
```

```
 If ( dsOL_LckScope = '1' ) ;
 MsgScope = 'THREAD' ;
 Else ;
 If ( dsOL_LckScope = '2' ) ;
 MsgScope = 'LOCK SCAPE' ;
 Else ;
 MsgScope = 'JOB' ;
 Endif ;
 Endif ;

 EXEC SQL EXECUTE cur1 USING :dsOL_JobName ,
 :dsOL_UserName , :dsOL_JobNbr , :dsOL_LckState ,
 :MsgLockStatus , :MsgLockType , :dsOL_Member ,
 :MsgShare , :MsgScope , :dsOL_ThreadID ;

 Endfor ;

 //*******************************************
 // Si demandé par le programme appelant,
 // génération d'un result set à partir de la
 // table temporaire
 //*******************************************
 If ( Resultset = 'YES' ) ;
 sql3 = 'SELECT distinct Job_name, Job_user_name, Job_number, ' +
 'Lock_state, Lock_status, Lock_type, Member_name, ' +
 'Share, Lock_scope ' +
 'FROM QTEMP/OBJL0100 FOR FETCH ONLY ' ;
 EXEC SQL
 PREPARE REQ1 FROM :sql3 ;
 EXEC SQL
 DECLARE C1 CURSOR FOR REQ1 ;
 EXEC SQL
 OPEN C1 ;
 EXEC SQL
 SET RESULT SETS CURSOR C1 ;

 Endif ;

 *inlr = *on ;

 /end-free
```

Voici le code source de la procédure stockée DB2 destinée à encapsuler le programme RPG ci-dessus :

```
CREATE OR REPLACE PROCEDURE votre_librarie/APIOBJLCKP (
 IN OBJNAME CHAR(10) ,
 IN OBJLIB CHAR(10) ,
 IN OBJTYPE CHAR(10) ,
 IN MEMBER CHAR(10) ,
 IN RESULTSET CHAR(3) )
 DYNAMIC RESULT SETS 1
 LANGUAGE RPGLE
 SPECIFIC votre_librarie/APIOBJLCKP
```

```
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
EXTERNAL NAME 'votre_librarie/APIOBJLCK'
PARAMETER STYLE SQL ;

COMMENT ON SPECIFIC PROCEDURE votre_librarie/APIOBJLCKP
IS 'API QWCLOBJL -> QTEMP/OBJL0100 ou Result Set' ;
```

Grâce à cette procédure stockée, vous serez en mesure d'appeler le programme RPG au moyen d'une requête SQL telle que celle ci-dessous :

```
CALL votre_librarie/APIOBJLCKP ('votre_table_DB2', 'la_bib_de_la_table', '*FILE', '*ALL', 'YES') ;
```

Vous l'aurez sans doute compris à la lecture du code, les paramètre 1 à 4 sont strictement identiques aux paramètres de la commande WRKOBJLCK. Le 5ème paramètre permet de préciser si vous souhaitez obtenir un "result set" en sortie. Si vous précisez pour cette valeur autre chose que YES, alors le "result set" ne sera pas produit, et vous devrez exécuter une requête complémentaire de type "SELECT * FROM QTEMP/OBJL0100" pour pouvoir exploiter le résultat produit par l'appel de l'API.

Si vous souhaitez savoir comment afficher dans une page HTML, via un script PHP, les informations renvoyées par le "result set", je vous invite à lire l'article complémentaire que j'ai déposé sur mon blog GregPhpLab.com.

Post-scriptum :

Suggestion d'amélioration : vous pouvez faire évoluer le programme RPG, de manière à ce qu'il soit en mesure d'accepter aussi bien les noms longs que courts en entrée. Il faut dans ce cas qu'il soit en mesure d'aller rechercher dans QSYS2/SYSTABLES le nom court de la table considérée, avant de le passer à l'API QWCLOBJL.

Retrouvez l'auteur de cet article sur son blog : http://gregphplab.com