Récupérer les définitions de colonnes d'une requête SQL avec SQLDA

- Les articles -

Date de mise en ligne : jeudi 4 novembre 2004

Description :

L'instruction SQL DESCRIBE permet de récupérer les définition de zones (nom, type, longueur, nb de décimales) d'une requête SQL à l'intérieur d'une DS (qu'il vous faudra déclarer) appellée SQLDA.

Environnement iSeries

Programme de service pour gérer la SQLDA

La SQLDA est composée d'une en-tête de 16 octets contenant une structure décrivant les informations d'entête de SQLDA(nombre de colonnes, nombre pivot...). et de descriptions de colonnes de 80 octets pour les informations de chaque colonne.

La méthode courante d'utilisation de SQLDA :

Exécuter une fois l'ordre SQL DESCRIBE en plaçant le nombre pivot à 0 (SQLN = 0). Cette opération permet de récupérer le nombre exact de colonnes(zones) de la requête et d'allouer la mémoire nécessaire au pointeur qui référence la SQLDA.

Exécuter une 2ème fois l'ordre SQL DESCRIBE pour alimenter l'ensemble des descriptions de colonnes (SQLN = nbColonnes X 80 + 16 octets d'entête).

Ensuite on se positionne sur la description de la 1ère colonne (après l'octet 16), puis on se déplace à l'intérieur de la SQLDA de 80 en 80 pour récupérer la descrition de chaque colonne à l'intérieur d'une DS (SQLVAR) décrivant chaque colone.

La SQLDA peut ainsi permettre d'analiser une requête sans en connaitre la structure avant l'exécution.

L'exemple fournit :

SQLDA.RPGLE reçoit en paramètre une requête SQL et place dans un tableau le Type (A ou N) la longueur, le nb de décimales ainsi que le nom de chaque colonne composant cette requête. Si le nom de la colonne ne peut pas être déterminé (par ex SUBST(ZON1, 1, 1)), le N° de rang de la zone à l'intérieur de la requête est retourné (par ex : « SELECT CODE, SUBST(ZON,1 , 1) » retourne '00002 ')

Le programme ne gère pas de fetch (pour cela il faut utiliser les 2 pointeurs SQLDATA (pour les données) et SQLIND (pour les valeurs NULL).

Dans certains cas, les descriptions de colonnes peuvent dépasser 80 octets (les BLOBs par ex). L'exemple fournit traite les types Packed, Signed, Date, Time, Timestamp et Char.

Description SQLDA

sqldaid => Champ de type caractère codé sur 8 octets contenant la chaîne SQLDA, qui identifie la structure SQLDA.

sqldabc => Longueur de la DS SQLDA.

sqln => Nombre de colones à alimenter(o au 1er appel, puis initialiser avec sqld pour récupérer l'ensemble des descriptions de colones dans sqlvar).

sqld => Nombre de colonnes de la requête. Cette zone est défini par l'instruction DESCRIBE.

sql_var => Champ de 80 caractères contenant la description de chaque colone.

Code source du programme de service (V5R4)

Programme de service

Dans cet exemple (en free rpg) j'utilise, pour gérer les erreurs SQL, un programme de service "SRVERRSQL".

Voir article sur SRVERRSQL

```
// ----------------------------------------------------------------------------------------

// ----------------------------------------------------------------------------------------  //  Lecture SQLDA pour récupérer les nom de zones et leurs définition d'une requête SQL  //  Paramètre en entée
=> Requête SQL 'ex : SELECT * FROM FICHIER'  //  Nb maximum de zones=> 999  // ----------------------------------------------------------------------------------------------------  H copyright('Serge
GOMES 2005') nomain  H  H Option( *SrcStmt : *DebugIO)  /COPY PROTOTYPE,SGSQLDA  /COPY PROTOTYPE,SRVERRSQL  PSGSQLDA  B  EXPORT
D  PI  *  D Ptr_rqt  *  VALUE  d Nb_occurs  5i 0


// Sql Descriptor Area  D SQLDA  DS  based(pSQLDA)  D  SQLDAID  1  8A  D  SQLDABC  9  12B 0  D  SQLN  13  14B 0  D
SQLD  15  16B 0  D  SQL_VAR  80A  DIM(1)


D SQLVAR  DS  based(pSQLVAR)  D  SQLTYPE  1  2B 0  D  SQLLEN  3  4B 0  D  WLong  3  3  D  WDec  4  4  D
SQLRES  5  16A  D  SQLDATA  17  32*  D  SQLIND  33  48*  D  SQLNAMELEN  49  50B 0  D  SQLNAME  51  80A


// SQLDA pointers  D pSQLDA  s  *  D pSQLVAR  s  *


// SQLDA sizes  DnSQLDA  s  5i 0  DszSQLDA  s  10i 0


// Colone descrp.  D DSCOLONE  ds  inz  D  pLong  1  4b 0  D  aLong  4  4  D  pDec  5  8b 0  D  aDec  8  8  //
----------  // Variables de travail  D  DW_rqt  s  32740  varying based(Ptr_rqt)  DW_x  s  5i 0
DPtr_zon  s  *  Dmod_zon  ds  occurs(999) likeds(W_DS_ZON)  d  based(Ptr_zon)  Ddtalen  s  10i 0 inz  DPstop
s  n  inz(*off)  // ----------ENTRY----------  /FREE  Nb_occurs = 0 ;  // Déclaration curseur  exec sql DECLARE FILTRE
CURSOR FOR SLT ;  // Préparation instruction  exec sql PREPARE SLT FROM :W_rqt ;  if SQLCOD <> 0 ;  // Erreur SQL Appel gezstion erreur sans blocage  callp
SqlErreur(sqlca ;Pstop) ;  return *null ;  endIf ;  // Lecture initiale SQLDA pour déterminer le nb de zones  (on met SQLN à 0)  szSQLDA = 16 ;  pSQLDA =
%alloc(szSQLDA) ;  SQLDA = *LOVAL ;  SQLN = 0 ;  // description instruction  exec sql DESCRIBE SLT INTO :SQLDA ;  if SQLN <= SQLD ;  nSQLDA = SQLD ;
szSQLDA = (nSQLDA * 80) + 16 ;  pSQLDA = %alloc(szSQLDA) ;  SQLN = nSQLDA ;  // description instruction (maintenat SQLN contient le nombre de zones de la requête)
  exec sql DESCRIBE SLT INTO :SQLDA ;  endIf ;  if SQLD > 999 ;  // Erreur Limite 999 zones atteinte  RETURN *null ;  endIf ;  // Lecture 1ère occurence requête
SQL dans SQLDA  pSQLVAR = %addr(SQL_VAR) ;  Nb_occurs = SQLD ;  dtaLen = %size(W_DS_ZON) * SQLD ;  Ptr_zon = %alloc(dtalen) ;  FOR W_x = 1 to SQLD ;
  %occur(mod_zon:W_x) ;  // détermine le type de la zone en cours  W_DS_ZON = *LOVAL ;  W_DEC  = 0 ;  W_TYP = 'A' ;  SELECT ;  // Packed (484-485)
Signed(488-489)  when SQLTYPE = 484 or SQLTYPE = 485 or  SQLTYPE = 488 or SQLTYPE = 489 ;  W_TYP = 'N' ;  if WDec <> x'00' ;  aDec = WDec ;
  W_DEC = pDec ;  endIf ;  aLong = WLong ;  W_LNG = pLong ;  // Date (384-385)  when SQLTYPE = 384 or SQLTYPE = 385 ;  W_LNG = 10 ;  //
Time (388-389)  when SQLTYPE = 388 or SQLTYPE = 389 ;  W_LNG = 10 ;  // Timestamp (392-393)  when SQLTYPE = 392 or SQLTYPE = 393 ;  W_LNG = 18 ;
  other ;  // char  W_LNG = SQLLEN ;  endsl ;  // Ecriture dans le tableau  W_NOM  = SQLNAME ;  mod_zon = W_DS_ZON ;  // Lecture occurence
suivante de SQLDA  eval pSQLVAR = pSQLVAR + 80 ;  EndFor ;  return Ptr_zon ;  /END-FREE  PSGSQLDA  E  //
----------
```

Prototype

```
* ?---------- ?*  * Protoype pour module SGSQLDA  *  *  Réalisation : Serge
GOMES  *  * ?---------- ?*  /IF DEFINED(SGSQLDA)
  /EOF  /ENDIF  /DEFINE SGSQLDA
  DSGSQLDA  PR  *  D Ptr_rqt  *  VALUE  d Nb_occurs
  5i 0

DW_DS_ZON  ds  89  D W_TYP  1  1  D W_LNG  2  5b 0
  D W_DEC  6  9b 0  D W_NOM  10  89
```

Code source du programme (V5R2

programme RPG

```
* ----------Euros*     * Lecture SQLDA pour récupérer les nom de zones et leurs définition d'une requête SQL     Euros*     *
Paramètre en entée => Requête SQL 'ex : SELECT * FROM FICHIER'                    Euros*     * Nb maximum de zones=> 80                                    Euros*
* ----------Euros*     H copyright('Serge GOMES 2004')
* ----------Euros*     *aSql Descriptor Area                                                                    Euros*
* ----------Euros*     D SQLDA         DS              based(pSQLDA)     D SQLDAID           1     8A     D SQLDABC           9
  12B 0     D SQLN              13     14B 0     D SQLD              15     16B 0     D SQL_VAR               80A   DIM(1)     D SQLVAR        DS              based(pSQLVAR)     D
SQLTYPE             1     2B 0     D SQLLEN            3     4B 0     D WLong             3     3     D WDec              4     4     D SQLRES            5     16A     D SQLDATA
17     32*     D SQLIND            33     48*     D SQLNAMELEN          49     50B 0     D SQLNAME           51     80A     *aSQLDA pointers     D pSQLDA        s           *     D
pSQLVAR         s           *     *aSQLDA sizes     DnSQLDA         s           5i 0     DszSQLDA        s           10i 0     *aColone descrp.     D DSCOLONE      ds              inz     D
pLong               1     4b 0     D aLong             4     4     D pDec              5     8b 0     D aDec              8     8
* ----------Euros*     *aVariables de travail     D     DW_X            s           5i 0     DP_RQT_SQL      s           9999
DW_RQT_SQL      s               LIKE(P_RQT_SQL)     DW_DS_ZON       ds          89     INZ     D W_TYP             1     1     D W_LNG             2     5b 0     D W_DEC             6
   9b 0     D W_NOM             10     89     DW_TB_ZON       s               DIM(80) LIKE(W_DS_ZON) INZ
* ----------ENTRY----------Euros*     C     *ENTRY       PLIST     C               PARM                P_RQT_SQL     C               EVAL
W_RQT_SQL = %TRIM(P_RQT_SQL)     *aDéclaration curseur     C/EXEC SQL     C+  DECLARE FILTRE CURSOR FOR SLT     C/END-EXEC     *aPréparation instruction     C/EXEC
SQL     C+  PREPARE SLT FROM :W_RQT_SQL     C/END-EXEC     C               IF      SQLCOD <> 0     *ÆErreur SQL     C               EVAL     *INLR = *ON     C
RETURN     C               ENDIF     *aLecture initiale SQLDA pour déterminer le nb de zones  (on met SQLN à 0)     C               EVAL     szSQLDA = 16     C               EVAL
pSQLDA = %alloc(szSQLDA)     C               EVAL     SQLDA = *LOVAL     C               EVAL     SQLN = 0     *adescription instruction     C/EXEC SQL     C+  DESCRIBE SLT INTO
:SQLDA     C/END-EXEC     C               IF      SQLN <= SQLD     C               EVAL     nSQLDA = SQLD     C               EVAL     szSQLDA = (nSQLDA * 80) + 16     C
EVAL     pSQLDA = %alloc(szSQLDA)     C               EVAL     SQLN = nSQLDA     *adescription instruction (maintenat SQLN contient le nombre de zones de la requête)     C/EXEC SQL
   C+  DESCRIBE SLT INTO :SQLDA     C/END-EXEC     C               ENDIF     C               IF      SQLD > 80     *ÆErreur Limite 80 zones atteinte     C               EVAL     *INLR =
*ON     C               RETURN     C               ENDIF     *aLecture 1ère occurence requête SQL dans SQLDA     C               EVAL     pSQLVAR = %addr(SQL_VAR)     C
FOR     W_X = 1 to SQLD     *adétermine le type de la zone en cours     C               EVAL     W_DS_ZON = *LOVAL     C               EVAL     W_DEC   = 0     C               EVAL
W_TYP = 'A'     C               SELECT     C               WHEN     SQLTYPE = 484 or SQLTYPE = 485 or          Packed     C                    SQLTYPE = 488 or SQLTYPE = 489
 Signed     C               EVAL     W_TYP = 'N'     C               IF      WDec <> x'00'     C               EVAL     aDec = WDec     C               EVAL     W_DEC = pDec     C
ENDIF     C               EVAL     aLong = WLong     C               EVAL     W_LNG = pLong     C               when     SQLTYPE = 384 or SQLTYPE = 385               Date     C
EVAL     W_LNG = 10     C               when     SQLTYPE = 388 or SQLTYPE = 389               Time     C               EVAL     W_LNG = 10     C               when     SQLTYPE = 392 or
SQLTYPE = 393           Timestamp     C               EVAL     W_LNG = 18     C               other                                Char     C               EVAL     W_LNG = SQLLEN     C
          endsl     *aEcriture dans le tableau     C               EVAL     W_NOM   = SQLNAME     C               EVAL     W_TB_ZON(W_X) = W_DS_ZON     *aLecture occurence suivante
de SQLDA     C               EVAL     pSQLVAR = pSQLVAR + 80     C               ENDFOR     C               EVAL     *INLR = *ON     C               return
```

* ----------Euros*