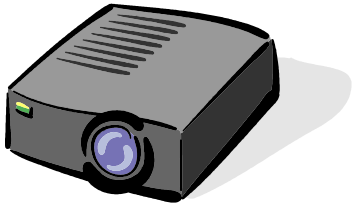
Modernisation, développement d'applications et DB2 sous IBM i
Technologies, outils et nouveautés 2013-2014

13 et 14 mai 2014 – IBM Client Center Paris, Bois-Colombes

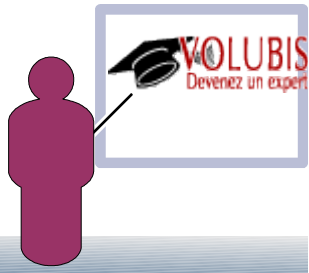


Volubis.fr

Conseil et formation sur OS/400, I5/OS puis IBM *i* depuis 1994 !

Dans nos locaux, vos locaux ou par Internet

Christian Massé - cmasse@volubis.fr



Fonctions SQL

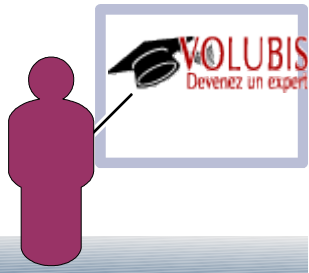
SQL propose de créer vos propres fonctions

-> UDF ou User Defined Fonction.

Ces fonctions qui permettent d'enrichir le SQL ne retournent qu'une valeur sous la forme date2 = findemois(date1)

Par exemple

```
Select datcde, datliv, findemois(datliv)
      from entcdes

Select * from entcdes
      Where findemois(datliv) > current date

Insert into entcdes (nocde, datcde, datliv)
      VALUES( (select max(nocde) + 1 from entcdes) ,
                  current date, findemois(current date)
                )
```



Fonctions SQL

SQL propose de créer vos propres fonctions

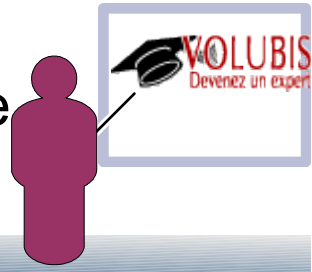
-> UDF ou User Defined Fonction.

Ces fonctions peuvent être écrites

- en SQL PSM (PL/SQL)
- en java (voir plus loin dans cette présentation)
- En langage RPG ou COBOL, auquel cas

Cela peut faire référence

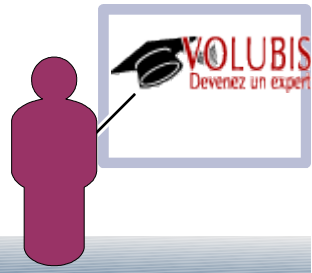
à un programme
à une procédure dans un programme de service



Fonctions SQL

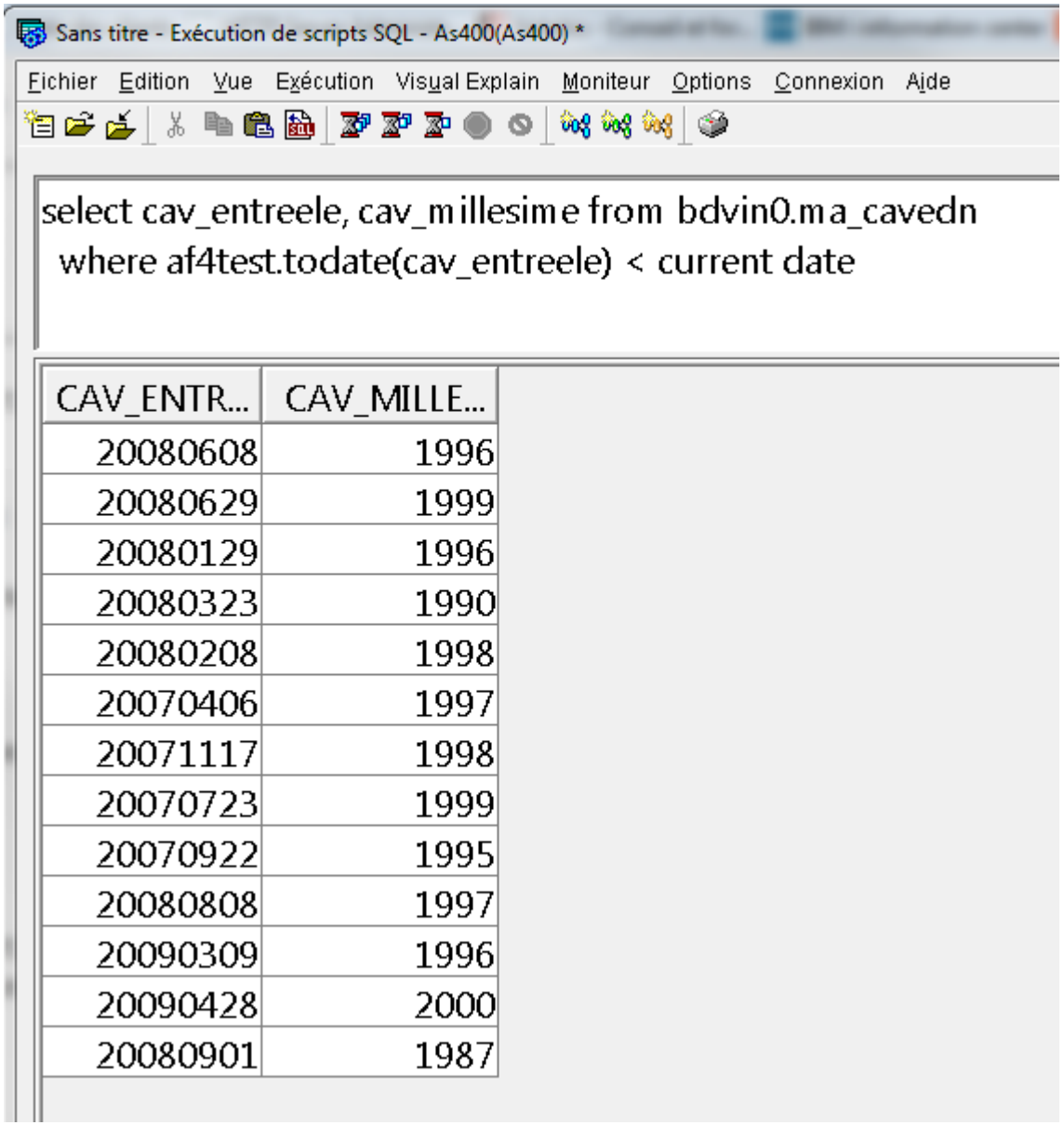
Exemple en SQL/PSM

```
CREATE FUNCTION AF4TEST/TODATE (
      DATIN DECIMAL(8, 0) )
      RETURNS DATE
      LANGUAGE SQL
BEGIN
RETURN
 DATE ( LEFT ( DATIN , 4 ) CONCAT '-'
  CONCAT SUBSTR ( DATIN , 5 , 2 ) CONCAT '-'
  CONCAT RIGHT ( DATIN , 2 ) ) ;
END  ;
```

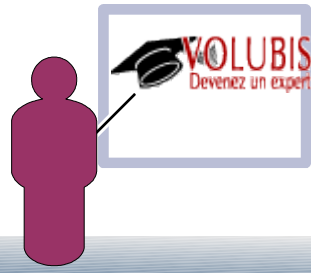


Fonctions SQL

Utilisation



CAV_ENTR...	CAV_MILLE...
20080608	1996
20080629	1999
20080129	1996
20080323	1990
20080208	1998
20070406	1997
20071117	1998
20070723	1999
20070922	1995
20080808	1997
20090309	1996
20090428	2000
20080901	1987

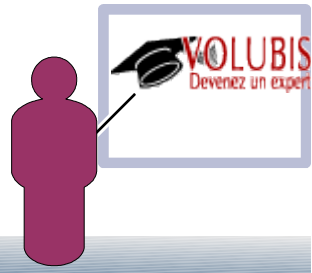


Fonctions SQL

Exemple en SQL/PSM

```
CREATE FUNCTION AF4TEST/NBRDEVINS (
     CODE INTEGER )
     RETURNS INTEGER
     LANGUAGE SQL
BEGIN
DECLARE NOMBRE INTEGER ;
SELECT COUNT ( * ) INTO NOMBRE
       FROM BDVIN0 / VINS WHERE PR_CODE = PROD ;
RETURN NOMBRE ;
END  ;
```

```
> select * from producteurs
    where nbrdevins(pr_code) = 4
  Instruction SELECT exécutée.
```



Fonctions SQL

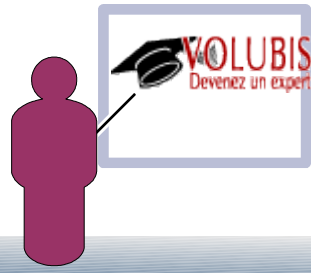
Exemple avec un programme de service :

```
CREATE FUNCTION AF4TEST/NBCAR (CHAR(50) )
                    RETURNS  INTEGER
      EXTERNAL NAME 'AF4TEST/NBCAR(NBCAR)'
      PARAMETER STYLE GENERAL RETURNS NULL ON NULL INPUT
```

RAPPEL : un programme de service est l'équivalent d'une DLL Windows.

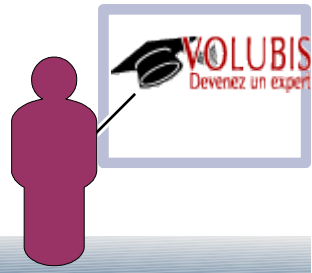
c'est à dire un objet contenant PLUSIEURS routines autonomes, créé à partir du langage C, COBOL ou RPG4.

Pour le GAP4, le source commence par **NOMAIN** (pas de pgm principal) et chaque routine est encadrées de deux "spécifs" **P** (début/fin) ou Dcl-proc/End-proc



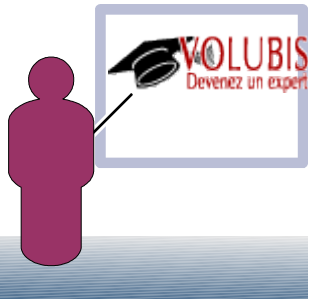
Fonctions SQL

```
 ****************************************************
 *  compte le nbr de caractères non blancs
 ****************************************************
H nomain

D nbcar           pr              10I 0
D  chaine                         50    const

Pnbcar            B                     export
D nbcar           pi              10I 0
D  chaine                         50    const

D w               s               10I 0
D i               s               10I 0
 /free
       for  i = 1 to %len(%trim(chaine));
         if  %subst(chaine : i : 1) <> ' ';
           w = w +1;
         endif;
       endfor;
    return  w;
 /end-free
Pnbcar            E
```



Fonctions SQL

```
////////////////////////////////////////////////////////////////
// free format : compte le nbr de caractères non blancs
////////////////////////////////////////////////////////////////
ctl-opt NOMAIN;

dcl-pr nbcar int(10);
    chaine char(50) const;
end-pr;

dcl-proc nbcar EXPORT;
  dcl-pi *N int(10);
    chaine char(50) const;
  end-pi;
  dcl-s w int(10);
  dcl-s i int(10);

    for  i = 1 to %len(%trim(chaine));
      if  %subst(chaine : i : 1) <> ' ';
         w = w +1;
      endif;
    endfor;
    return  w;
end-proc;
```



Fonctions SQL

Exemple avec un programme de service :

```
                        Entrée d'instructions SQL

Saisissez l'instruction SQL, puis appuyez sur ENTREE.
   > CREATE FUNCTION AF4TEST/NBCAR (CHAR(50) )
                               RETURNS  INTEGER
            EXTERNAL NAME 'AF4TEST/NBCARF(NBCAR)'
            PARAMETER STYLE GENERAL RETURNS NULL ON NULL INPUT
     La fonction NBCAR a été créée dans AF4TEST.
```

```
                        Entrée d'instructions SQL

Saisissez l'instruction SQL, puis appuyez sur ENTREE.
===> values nbcar(char('bonjour tout le monde'))
```

```
....+....1....
VALUES
18
```

```
select * from bdvin1/producteurs where nbcar(pr_nom) < 10
```

PR_CODE	PR_NOM
253	Petrus
288	Auvigue
295	Verget
464	Merlin

Fonctions SQL

Exemple avec un programme :

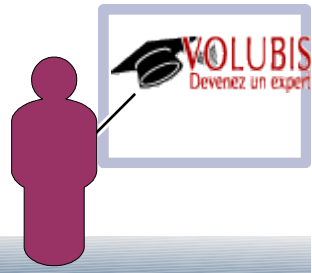
```
CREATE FUNCTION AF4TEST/NBNUM (CHAR(50) )
                    RETURNS  INTEGER
      EXTERNAL NAME 'AF4TEST/NBNUM'
      LANGUAGE RPGLE NO SQL
      PARAMETER STYLE SQL RETURNS NULL ON NULL INPUT
```

Dans le cas d'un programme (tout langage), vous recevez des paramètres de manière traditionnelle. Autant de paramètres qu'il y en a dans la fonction, plus UN pour la valeur retour.

:

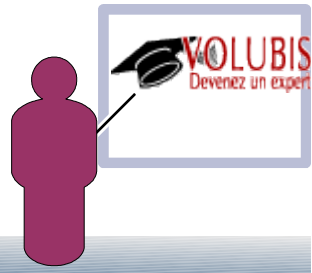
:

Ainsi que ceux correspondant à votre choix (SQL, DB2SQL, ...)



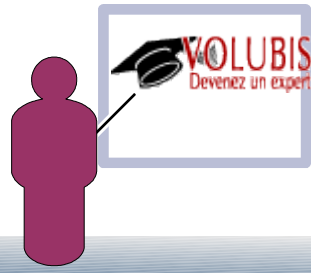
Fonctions SQL

```
 ****************************************************************
 * compte le nbr de chiffre avant premier blanc
 ****************************************************************

 *paramètres
DNBNUM            PI                              EXTPGM('NBNUM')
D chaine                          50
D retour                          10I 0
 *indicateurs SQL (-1 = null, 0 = non null)
D chaine_ind                       5I 0
D retour_ind                                      like(chaine_ind)
 *
D SQLSTATE                         5
D fonction_qual                  139
D fonction_nom                   128
D msg_diag                        70               varying
 * autres variables de travail
D w               s               10I 0
D i               s               10I 0
```



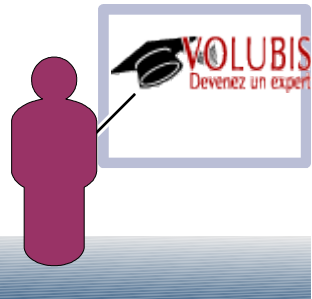
Fonctions SQL

```
/free
  // début du code
   SQLSTATE = '00000';
   for  i = 1 to %len(%trim(chaine));
     if  %subst(chaine : i : 1) < '0'
                    or %subst(chaine : i : 1) > '9';

     // car autre que blanc ==> erreur
        if  %subst(chaine : i : 1) <> ' ';
            SQLSTATE = '38I01';
            msg_diag = 'zone contient des car. invalides';
        endif;

      Else;
           // un chiffre
            w = w +1;
      endif;
    endfor;

  retour = w;
  *inlr = *on;
/end-free
```



Fonctions SQL

```
////////////////////////////////////////////////////////////
// free Form, compte le nbr de chiffre avant premier blanc
////////////////////////////////////////////////////////////
 ctl-opt dftactgrp(*no);
 // paramètres
 dcl-pi *N;
    chaine char(50);
    retour int(10);
    chaine_ind int(5);
    retour_ind like(chaine_ind);
    SQLSTATE char(5);
    fonction_qual char(139);
    fonction_nom char(128);
    msg_diag varchar(70);
 end-pi;
 dcl-s w int(10);
 dcl-s i int(10);
  // début du code
    SQLSTATE = '00000';

      …
```



Fonctions SQL

Exemple avec un programme

```
CREATE FUNCTION AF4TEST/NBNUM (CHAR(50) )
                            RETURNS  INTEGER
       EXTERNAL NAME 'AF4TEST/NBNUMF'
       LANGUAGE RPGLE NO SQL
       PARAMETER STYLE SQL RETURNS NULL ON NULL INPUT
La fonction NBNUM a été créée dans AF4TEST.
```

```
 values nbnum(char('12345678901    '))
```

```
VALUES
11
```

```
values nbnum(char('123x5678901    '))
L'analyse ne peut être exécutée.  Consultez les messages
```

```
Erreur de fonction définie par l'utilisateur sur le membre QSQPTABL.
Réponse d'annulation reçue pour message CPF503E.
```

F1

```
  1 -- Le programme externe ou le programme de service a renvoyé SQLSTATE
38I01. Le message renvoyé par le programme est : zone contient des car.
```

Fonctions SQL

Il peut exister plusieurs versions d'une même fonction

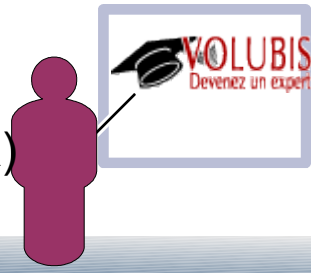
par exemple CALECHEANCE(date) et CALECHEANCE(dec(8,0)),
dans le fichier SYSFUNCS, cela s'appelle une signature (PARM_SIGNATURE)

c'est le type qui fait référence, pas la longueur
(par exemple transmettre dec(8,0) ou dec(6,0) c'est pareil)

Donc si vous essayez

> SELECT NBNUM('123456') from QSQPTABL
NBNUM de type *N dans *LIBL non trouvé. (SQL0204)

SQL n'a pas trouvé la version de NBNUM attendant du VARCHAR
(en effet les constantes sont toujours traitées par SQL comme du VARCHAR)



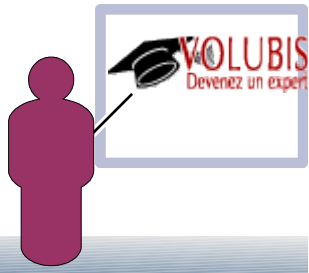
Fonctions SQL

le plus simple semble être de créer une deuxième version de la fonction :

```
CREATE FUNCTION AF4TEST/NBNUM (p1 VARCHAR(50) )
              RETURNS INTEGER
LANGUAGE SQL
RETURN NBNUM( CAST(P1 as CHAR(50) )  )
```

```
> CREATE FUNCTION AF4TEST/NBNUM (p1 VARCHAR(50) )
                  RETURNS INTEGER
  LANGUAGE SQL
  RETURN NBNUM( CAST(P1 as CHAR(50) ) )
  La fonction NBNUM a été créée dans AF4TEST.

   values nbnum('12345678901    ')
```

```
VALUES
11
```



Fonctions SQL

L'utilisation de fonctions a un coût car on ne peux pas utiliser directement d'index.

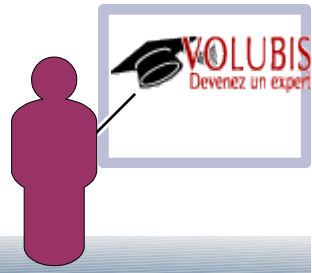
les fonctions écrites en RPG, particulièrement dans des *SRVPGM, sont plus rapides.

Celles écrites en Java, probablement les plus longues à l'exécution.

Pour améliorer les performances, ajoutez :

- NOT FENCED

  sans ce paramètre la fonction est lancée dans un second JOB de type BCI
  (ce qui d'ailleurs rend compliqué le Debug)



Fonctions SQL

L'utilisation de fonctions a un coût car on ne peux pas utiliser directement d'index.

Pour améliorer les performances, ajoutez :

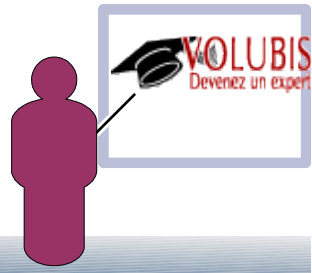
- DETERMINISTIC

  indiquant que cette fonction avec les mêmes paramètres fournie toujours les mêmes résultats

- NO EXTERNAL ACTION

  indiquant que cette fonction ne lance pas d'autres traitements que DB2 (Data area, IFS, etc...)

Ces deux derniers paramètres vont permettre à DB2 de faire du cache et de ne jamais appeler deux fois la fonction avec des valeurs identiques



Fonctions SQL

Quelques remarques pour les fonctions écrites en Java *(HTTPGetBlob, par exemple)* :

les classes Java doivent être copiées dans /QIBM/UserData/OS400/SQLLib/Function

les fichiers jar doivent être en enregistrés par la procédure SQLJ.INSTALL_JAR

la connexion peut se faire avec les paramètres **jdbc:default:connection** pour se connecter à la base locale

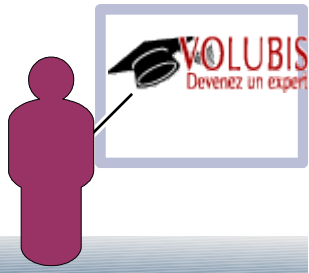
l'accès aux données se fait en Unicode, donc pas de JOB au CCSID à 65535 !

la méthode doit être public static (void pour une procédure, un type retour compatible SQL, pour une fonction)

les types des paramètres doivent être compatibles entre Java et SQL**,**

Exemple :

```
CREATE FUNCTION SYSTOOLS.HTTPGETBLOB (
        URL VARCHAR(2048) CCSID 1208 ,
        HTTPHEADER CLOB(10240) CCSID 1208 )
        RETURNS BLOB(2147483647)
        LANGUAGE JAVA
        SPECIFIC SYSTOOLS.HTTPGETBLOBNONXML
        NOT DETERMINISTIC
        NO SQL
        CALLED ON NULL INPUT
        EXTERNAL NAME
             'SYSTOOLS.DB2RESTUDF:com.ibm.db2.rest.DB2UDFWrapper.httpGetBlob'
        PARAMETER STYLE JAVA ;
```

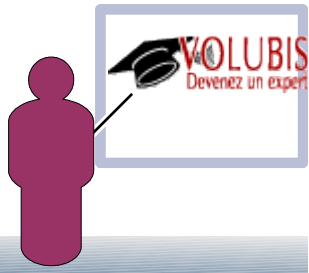


Fonctions TABLE

Pour retourner plusieurs valeurs, vous pouvez écrire des fonctions tables, qui retourne des lignes (donc une série de colonnes)
comme une table (ou un fichier)

Exemple :

```
CREATE FUNCTION SYSTOOLS/HTTPGETBLOBVERBOSE (
        URL VARCHAR(2048) CCSID 1208 ,
        HTTPHEADER CLOB(10240) CCSID 1208 )
        RETURNS TABLE (
         RESPONSEMSG BLOB(2147483647) ,
         RESPONSEHTTPHEADER CLOB(1048576) CCSID 1208 )
        LANGUAGE JAVA
        SPECIFIC SYSTOOLS/HTTPGETBLOBVERBOSENONXML
        NOT DETERMINISTIC
        NO SQL
        CALLED ON NULL INPUT
        SCRATCHPAD 100
        EXTERNAL NAME
            'SYSTOOLS.DB2RESTUDF:com.ibm.db2.rest.DB2UDFWrapper.httpGetBlobVerbose'
        PARAMETER STYLE DB2GENERAL ;
```



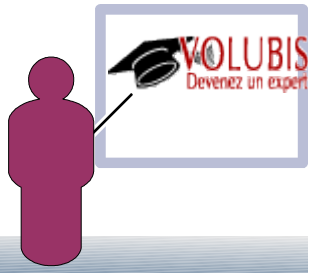
Fonctions TABLE

1/ en PL SQL

cela consiste à placer un SELECT sur la clause RETURN.

IBM fourni un exemple avec la fonction USERS() dont voici le source

```
CREATE FUNCTION SYSIBM.USERS ( )
 RETURNS TABLE (
   ODOBNM CHAR(10),
   ODOBTX CHAR(50) )
 LANGUAGE SQL
 READS SQL DATA

BEGIN
 DECLARE CMD CHAR ( 300 ) DEFAULT ' ' ;
 DECLARE WARN CONDITION FOR '01HII' ;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SIGNAL WARN SET MESSAGE_TEXT = 'SOME
     USERS NOT AVAILABLE' ;
 SET CMD = 'QSYS/DSPOBJD qsys/*ALL *USRPRF OUTPUT(*OUTFILE) ' CONCAT ' OUTFILE(qtemp/q_users) ' ;
 CALL QSYS . QCMDEXC ( CMD , 0000000300.00000 ) ;
 RETURN SELECT ODOBNM , ODOBTX FROM QTEMP.Q_USERS ;
END  ;
```



Fonctions TABLE

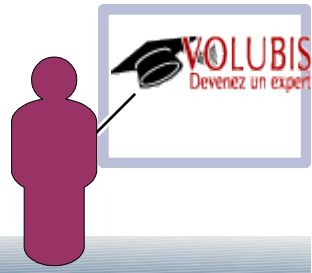
Utilisation :

SELECT * from **TABLE**(udtf()) as alias

→ Select * from TABLE (sysibm.users()) as u

```
Select * from table (sysibm.users() ) as u;
```

ODOBNM	ODOBTX
AF400	Profil propriétaire pour AF400
AF400CM	cmasse@volubis.fr



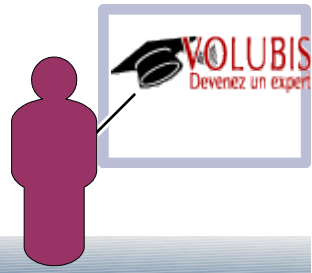
Fonctions TABLE

Utilisation :

Bien sur, vous pouvez utiliser TOUTE la puissance de la syntaxe SQL

```
Select * from table (sysibm.users() ) as u
  where ODOBNM like 'FORMA%'
```

ODOBNM	ODOBTX
FORMATION	profil de référence FORMATION
FORMATIO...	profil formation DIX
FORMATIO...	profil formation ONZE
FORMATIO...	Profil formation UN
FORMATIO...	Profil formation DEUX
FORMATIO...	Profil formation TROIS
FORMATIO...	Profil formation QUATRE
FORMATIO...	Profil formation CINQ
FORMATIO...	Profil formation SIX
FORMATIO...	Profil formation SEPT
FORMATIO...	profil formation HUIT
FORMATIO...	profil formation NEUF



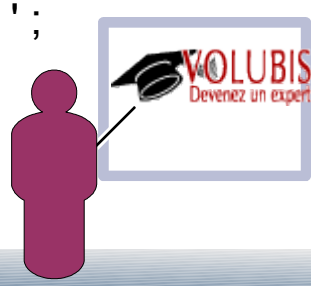
Fonctions TABLE

Dans ce cadre, nous vous donnons un exemple basé sur DSPPGMREF

```
CREATE FUNCTION FORMATION0.PGMREF (
 PGMPARM VARCHAR(10),
 LIBPARM VARCHAR(10)  )
 RETURNS TABLE (
   PGM CHAR(10),
   OBJRFF CHAR(11),
   LIBREF CHAR(11),
   OBJTYPE CHAR(10) )
 LANGUAGE SQL
 READS SQL DATA

BEGIN
 DECLARE CMD CHAR ( 300 ) DEFAULT ' ' ;
 DECLARE WARN CONDITION FOR '01HII' ;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SIGNAL WARN SET MESSAGE_TEXT =
        'Liste en erreur' ;
 SET CMD = 'QSYS/DSPPGMREF PGM(' CONCAT trim(LIBPARM) concat '/' CONCAT
                  trim(pgmparm) CONCAT ') OUTPUT(*OUTFILE) OUTFILE(qtemp/dsppgm) ' ;
 CALL QSYS . QCMDEXC ( CMD , 0000000300.00000 ) ;
 RETURN SELECT WHPNAM, WHFNAM, WHLNAM, WHOTYP FROM qtemp.dsppgm ;
END  ;
```

```
select * from table( formation0.pgmref('*ALL' , 'AF400') ) as p
```

PGM	OBJRFF	LIBREF	OBJTYPE
AFPVIEWCL	QUSRSPLA	*LIBL	*PGM
AFPVIEWCL	QLECWI	QSYS	*SRVPGM
AFPVIEWCL	QCLSRV	QSYS	*SRVPGM
AFPVIEWCL	QLEAWI	QSYS	*SRVPGM
ANZLOG	QSQROUTE	QSYS	*PGM
ANZLOG	ZMFI1	SPLF2	*FILE
ANZLOG	ANZLOGD	SPLF2	*FILE



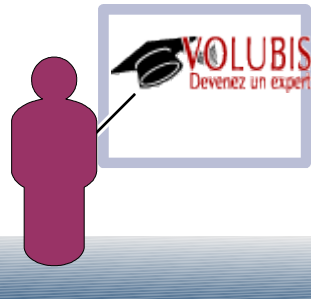
Fonctions TABLE

2/ External (ici en RPG utilisant des API)

Dans cet exemple, nous utiliserons l'API QUSLMBR qui permet d'obtenir la liste des membres dans un User Space

-> la fonction est définie de la manière suivante

```
CREATE FUNCTION AF4TEST/LISTMBR (
      BIB CHAR(10),
      FILE char(10) ,
      MBR char(10) )
 RETURNS  TABLE (NOM CHAR(10) , TYPE CHAR(10), DATCRT DATE ,
                 DATCHG DATE,   texte char(50)
                )
 EXTERNAL NAME 'AF4TEST/LISTMBR'
 PARAMETER STYLE DB2SQL
 DISALLOW PARALLEL
```



Fonctions TABLE

Paramètres en entrée :

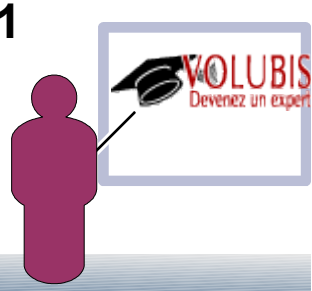
autant que de paramètres déclarés (ici 3)
+ autant que colonnes retour (ici 5)

des variables binaires (8) pour signaler la nullité(-1) ou pas (0) des paramètres

puis des paramètres standard SQL :

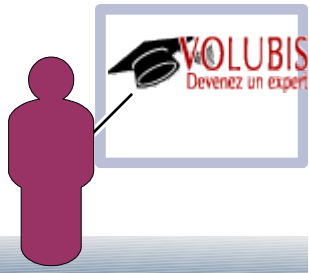
SQLSTATE
nom qualifié de la fonction
nom simple de la fonction
une zone message (taille variable)
call_type : type d'appel,le pgm est appelé

- une première fois pour réaliser un pseudo "Open", call_type est alors égal à **-1**
- *n* fois avec call_type à **0**, il doit retourner UNE ligne ou SQLSTATE à '02000' pour signaler la fin de fichier
- une dernière fois représentant un pseudo "Close" avec call_type à **1**



Fonctions TABLE

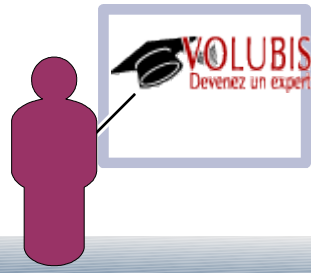
```
dLISTMBR          PI                  EXTPGM('LISTMBR')
 *paramètres définis dans CREATE FUNCTION
d INbib                          10
d INfile                         10
d INmbr                          10
 *paramètres définis dans CREATE FUNCTION
d  OUTnom                        10
d  OUTtype                       10
d  OUTdatcrt                       D
d  OUTdatchg                       D
d  OUTtexte                      50
 * indicateurs valeur nulle
d INbib_i                         5I 0
d INfile_i                        5I 0
d INmbr_i                         5I 0
 * indicateurs valeur nulle
d  OUTnom_i                       5I 0
d  OUTtype_i                      5I 0
d  OUTdatcrt_i                    5I 0
d  OUTdatchg_i                    5I 0
d  OUTtexte_i                     5I 0
 * paramètres standards (STYLE SQL)
DSQLSTATE                         5
Dfonction_qual                  517    VARYING
Dfonction_name                  128    VARYING
Dmessage_diag                    80    VARYING
Dcall_type                       10I 0
```



Fonctions TABLE

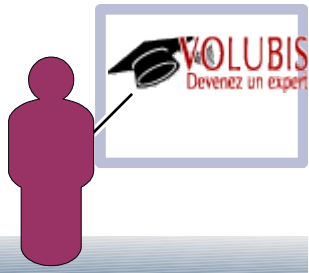
Autres déclarations

```
 * prototype pour API création/destruction User Space
dQUSCRTUS         PR                  EXTPGM('QUSCRTUS')
d  space                         20    CONST
d  USattribut                    50    CONST
d  UStaille                      10I 0 CONST
d  UScontenu                      1    CONST
d  USdroits                      10    CONST
d  UStexte                       50    CONST
d  USreplace                     10    CONST
d  USerrcode                           likeds(errcodeDS)
dQUSDLTUS         PR                  EXTPGM('QUSDLTUS')
d  space                         20    CONST
d  USerrcode                           likeds(errcodeDS)
 * prototype pour API qui retrouve pointeur de début
dQUSPTRUS         PR                  EXTPGM('QUSPTRUS')
d  space                         20    CONST
d  ptr                            *
Dusrspc           s                20   inz('LISTMBR   QTEMP')
dQUSLMBR          PR                  EXTPGM('QUSLMBR')
d  space                         20    CONST
d  format                         8    CONST
d  ficlib                        20    CONST
d  membre                        10    CONST
d  OVRDBF                         N    CONST
```



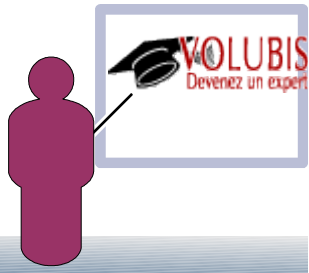
Fonctions TABLE

```
 * va contenir l'adresse de début du User Space
Dpointeur          s                 *
DI                 s               10i 0

 * l'entête
Dptrinfos          s                 *
DRTVINF            ds                      based(ptrinfos)
D  offset                          10i 0
D  taille                          10i 0
D  nbpostes                        10i 0
D  lgposte                         10i 0

 * la liste
dptrliste          s                 *
Dmembre            ds                      based(ptrliste) qualified
d  nom                             10
d  type                            10
d  DatCrt                           7
d  HeurCrt                          6
d  DatChg                           7
d  HeurChg                          6
d  texte                           50
DerrcodeDS         ds                      qualified
d  tailleDS                        10i 0 inz(%size(errcodeDS))
d  taille                          10i 0
d  msgID                            7
d  reserve                          1
d  errdta                          50
```



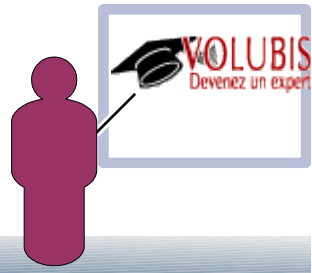
Fonctions TABLE

```
 if call_type < 0 ;
    // premier appel, constitution de la liste
    SQLSTATE = '00000'  ;

     // création du user Space dans QTEMP
     QUSCRTUS(usrspc: *Blanks: 1024: x'00': '*USE':
          'Liste des membres': '*YES' : errcodeDS);
     // remplissage avec la liste des membres
     QUSLMBR(usrspc: 'MBRL0200': INfile + INbib
        : INmbr: *OFF);

     // recupération et émmorisation pointeur de début
     QUSPTRUS(usrspc : pointeur);
     ptrinfos = pointeur + 124;

     // position sur 1er poste
     ptrliste = pointeur + offset;
    return;
elseif    call_type = 0 ;
    // appel "normal", retour d'un membre
    i+=1;
    if i<=nbpostes;
      // on est déjà sur le poste avec le pointeur
      OUTnom = membre.nom;
      OUTtype = membre.type;
```



Fonctions TABLE

```
        // test date non convertible (probablement à zéro)
        Monitor;
          OUTdatcrt = %date(membre.datcrt : *CYMD0);
         on-error *all;
          OUTdatcrt_i = -1;
        Endmon;
        Monitor;
          OUTdatchg = %date(membre.datchg : *CYMD0);
         on-error *all;
          OUTdatchg_i = -1;
        Endmon;

        OUTtexte = membre.texte;
        // position sur prochain poste (sauf dernière fois)
        if i<nbpostes;
          ptrliste = ptrliste + lgposte;
        endif;
        else;
        // fin de liste
        SQLSTATE = '02000';
        *INLR = *on;
      endif;
      return;
else;
    // appel final, faire le ménage
    QUSDLTUS(usrspc: errcodeDS) ;
    *inlr = *on;
endif;
```



Fonctions TABLE

Utilisation

Comme vu plus haut, toute constante étant réputée de type VARCHAR, il faut *caster*

```
                          Entrée d'instructions SQL

Saisissez l'instruction SQL, puis appuyez sur ENTREE.
===> select * from table( af4tool.listmbr(char('FORMATION0'),
                                          char('QRPGLESRC') ,
                                          char('*ALL')) ) as m
```

```
                          Affichage des données
                                          Largeur des données  . . :    94
Première ligne à afficher . . ______     Première colonne à afficher  . _____
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+..
NOM          TYPE         DATCRT    DATCHG    TEXTE
CLIENTTRG    RPGLE        13/02/02  13/02/02  Trigger / CLIENTP1, avec pointeurs
CM01T        RPGLE        20/11/12  20/11/12
DBLCARFCT    RPGLE        07/04/09  07/04/09  fonction DBLCAR, double un caract
ERRNO_H      RPGLE        30/05/07  30/04/07  ERRNO_H, (C) Scott C. Klement
FREE00       RPGLE        30/04/09  30/04/09  recherche et remplacement d'une c
FREE01       RPGLE        30/04/09  30/04/09  ILE, TP1 retoune N° jour d'une DA
FREE01FCT    RPGLE        04/05/09  04/05/09  ILE, TP1 sous forme de fonction
FREE02       RPGLE        30/04/09  30/04/09  ILE, TP2 retourne fin de mois d'u
```

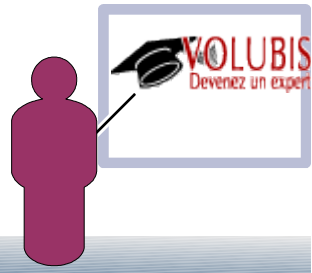
Fonctions TABLE

Utilisation

Là aussi la fonction aurait pu être utilisée comme ceci

```
select * from table( af4tool.listmbr(char('AF4SRCT'),
                                     char('EXEMPLEILE') ,
                                     char('*ALL')) ) as m
where year(datcrt) = year(current date)
 and  month(datcrt) = month(current date)
```

```
                          Affichage des données
                                                Largeur des données  . . :    94
Première ligne à afficher . . ______      Première colonne à afficher  . _____
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
NOM          TYPE        DATCRT    DATCHG    TEXTE
LISTMBR      RPGLE       25/07/13  29/07/13  Liste les membres d'un fichier
LISTMBRSQL   SQL         25/07/13  25/07/13  Liste les membres (CREATE FUNCTION
********  Fin de données  ********
```



Fonctions TABLE

Quelques fonctions tables fournies par IBM

GET_JOB_INFO()	infos sur un JOB
OBJECT_STATISTICS()	liste d'objets
DISPLAY_JOURNAL()	affiche le contenu du journal
USERS	liste les utilisateurs
GROUP_USERS	les membres d'un groupe

Plus toutes les fonctions Httpxxxx qui sont livrées sous deux formes

UDF retournant le flux fourni par le serveur http interrogé
UDTF retournant deux colonnes :
1/ le flux
2/ l'entête http (contenant le code 404, par exemple)

HttpGetBlobVerbose	httpGetClobVerbose
HttpPutBlobVerbose	httpPutClobVerbose
HttpPostBlobVerbose	httpPostClobVerbose
HttpDeleteBlobVerbose	httpDeleteClobVerbose
HttpBlobVerbose	httpClobVerbose
httpHeadVerbose