Université IBM i 2018

16 et 17 mai

IBM Client Center Paris

S49 – Accès dynamique aux données DB2 avec les UDTFs

Philippe Bourgeois
IBM France
pbourgeois@fr.ibm.com

Plan de la présentation

- 1. Introduction aux routines SQL
- 2. Les UDTFs SQL
- 3. Exploitation
- 4. UDTFs fournies par IBM
- Annexe : les UDTFs externes

1. Introduction aux routines SQL

Routines SQL

- Code exécutable encapsulé dans un "objet SQL" :
  - **Procédures cataloguées**
    - Stored procedures
    - Programmes en SQL appelables par un CALL SQL
    - Créées par l'instruction SQL *CREATE PROCEDURE*
  - **Fonctions**
    - UDF – User Defined Functions et **UDTF – User Defined Table Functions**
    - Création de ses propres fonctions SQL
    - Créées par l'instruction SQL *CREATE FUNCTION*
  - **Déclencheurs**
    - Triggers
    - Programme appelé sur des opérations d'insertion, mise-à-jour ou suppression d’enregistrements
    - Créés par l'instruction SQL *CREATE TRIGGER*

Routines SQL

- Le langage utilisé pour développer des routines SQL peut être :
  - Le langage SQL
    - On parle alors de SQL/PSM ou SQL procédural
  - Un autre langage (RPG/COBOL/C/C++, Java, REXX)
    - Les routines sont dites externes
    - Elles encapsulent un programme ou une procédure (RPG, COBOL…)
  - La création de routines alimente le catalogue SQL :
    - SYSROUTINES, SYSTRIGGERS, SYSPROCS, SYSFUNCS, SYSPARMS
  - La création de routines génère un programme ou une procédure de type CLE (ILE C)

Langage SQL/PSM

- Le langage SQL/PSM est constitué d'instructions :
  - 1. De début et fin de procédure
    - `BEGIN et END`
  - 2. De déclaration de variables
    - `DECLARE nom_variable type(longueur)`
  - 3. De déclaration de conditions et gestionnaires d'erreur
    - `DECLARE` *`variable`* `CONDITION FOR` *`code_erreur`*
    - `DECLARE CONTINUE/UNDO/EXIT HANDLER FOR` *`condition instruction`*
  - 4. D'alimentation de variables
    - `SET et VALUES`

Langage SQL/PSM

- Le langage SQL/PSM est constitué d'instructions :
  - 5. De contrôle
    - `IF, FOR, REPEAT, WHILE, CASE, GOTO, LOOP, ITERATE, LEAVE, RETURN`
  - 6. De gestion des erreurs
    - `SIGNAL, RESIGNAL, GET DIAGNOSTICS`
  - 7. De toute autre instruction SQL
    - `SELECT, INSERT, UPDATE...`
    - `DECLARE CURSOR, FETCH...`
    - `CALL`
    - `...`

Fonctions SQL

- **UDF** – User Defined Function
  - Ne retourne qu'une seule valeur
  - 2 types :
    - SQL : création de sa propre fonction scalaire en SQL/PSM
    - Externe : utilisation d'un programme ou d'une procédure de programme de service IBM i pour se créer sa propre fonction scalaire

- **UDTF** – User Defined *Table* Function
  - Retourne un ensemble de lignes-colonnes
  - 2 types : SQL ou externe

Exemple d'UDF – 1/2

TABEMPL

MAT	NOM	SX	SRV	SAL	DAT_NAI
20	MICHEL	M	911	2140,00	1955-03-17
10	ANNIE	F	911	3080,00	1959-05-12
50	JACQUES	M	911	2360,00	1956-11-11
40	DANIELE	F	977	2810,00	1962-07-13
30	MARC	M	977	3570,00	1972-02-28
60	CLAUDE	M	990	2210,00	1988-05-02
70	RICHARD	M	-	-	1956-03-01

TABSERV

SRV	NOMSRV
901	COMPTA
977	MANUFACT
911	VENTES
982	APRES VTES

```
19  -- Quels sont les services avec plus de 2 employés ?
20  SELECT * FROM tabserv WHERE nbemp(srv) >=2;
```

SRV	NOMSRV
977	MANUFACT
911	VENTES

Exemple d'UDF – 2/2

```
CREATE OR REPLACE FUNCTION nbemp (srvin INT) RETURNS INT
LANGUAGE SQL
BEGIN
 DECLARE nb INT;
 SELECT COUNT(*) INTO nb FROM tabempl WHERE srv = srvin;
 RETURN nb;
END;
```

```
Objet       Type       Biblio      Attribut
NBEMP       *SRVPGM    AS425F      CLE
```

2. Les UDTFs SQL

Les UDTFs

- UDTF : User Defined **Table** Function
- Fonction qui reçoit des paramètres en entrée et qui retourne une "table" : un ensemble de lignes-colonnes
  - On les appelle également les Fonctions Table
- Cette table "dynamique" peut être ensuite interrogée comme n'importe quelle autre table (clause FROM du SELECT)
- 2 types : SQL ou externe
  - SQL : le code source est en SQL/PSM
  - Externe : utilisation d'un programme ou d'une procédure IBM i
  - Dans les 2 cas création par l'ordre SQL `CREATE FUNCTION`

Les UDTFs

- Les UDTFs permettent de construire un ensemble dynamique de données
  - Les données en question peuvent provenir de tables DB2… ou d'ailleurs (appel de Services Web, lecture de fichiers IFS, données calculées, appel de procédures…)
- Elles peuvent être utilisées depuis toutes les interfaces SQL
  - ODBC, JDBC, .NET, OLE DB, CLI, PHP…
  - Et bien sûr nativement sous IBM i (SQLRPGLE, SQLCBLLE…)
- Elles sont utilisées en interne IBM ainsi que pour certains "Services IBM i" et "Services DB2" (voir point 4)

Les types d'UDTFs

- UDTFs **SQL**
  - Entièrement écrites en SQL
    - Instruction CREATE FUNCTION pour la créer
    - Code en SQL/PSM
  - Les données peuvent être retournées
    - Soit par un RETURN d'un SELECT (1 ou plusieurs lignes)
    - Soit par l'instruction PIPE (1 ligne à la fois)

- UDTFs **externes**
  - Instruction CREATE FUNCTION pour la créer
  - Code externe (appel d'un programme ou d'une procédure de programme de service)

Création d'une UDTF

- Ordre de création :

```
CREATE <OR REPLACE> FUNCTION nom_fonction
    < (déclaration des paramètres) >
    RETURNS TABLE (description des colonnes retournées)
    <Options>
```

SQL

<BEGIN> corps de la fonction <END>

OU

Externe

<EXTERNAL NAME nom du programme ou de la procédure IBM i>

La déclaration des paramètres

- Syntaxe : `nom_du_paramètre  type_de_paramètre(<longueur>)`
  `<DEFAULT valeur_par_défaut>`
- Clause DEFAULT : depuis la version 7.2
- Exemple
  - `CREATE FUNCTION f1 (z1 INT, z2 CHAR(2) DEFAULT('92'), z3 DATE DEFAULT CURRENT DATE) RETURNS TABLE(...)`
  - L'appel à la fonction peut alors se faire de différentes façons :
    - `f1(10, '78', '2018-05-17')`
    - `f1(10, DEFAULT, '2018-05-17')`
    - `f1(10, z3=>'2018-05-17')`
    - `f1(10, '78')`
    - `f1(10)`

La description des colonnes retournées

- Clause RETURNS TABLE

- Syntaxe : `RETURNS TABLE(nom_de_la_colonne type_de_la_colonne(<longueur>)`

```
CREATE OR REPLACE FUNCTION udtf1 (nosrv  SMALLINT)
RETURNS TABLE (srv SMALLINT, nom CHAR(12), nomsrv CHAR(10))
```

UDTF SQL – Retour avec RETURN d'un SELECT

```
866 CREATE OR REPLACE FUNCTION udtf1 (nosrv  SMALLINT)
867 RETURNS TABLE (srv SMALLINT, nom CHAR(12), nomsrv CHAR(10))
868 LANGUAGE SQL READS SQL DATA
869 RETURN
870  SELECT srv, nom, nomsrv FROM tabserv JOIN tabempl
871  USING(srv) WHERE srv=nosrv;
872
873 SELECT * FROM TABLE(udtf1(977)) AS srvemp;
```

SRV	NOM	NOMSRV
977	DANIELE	MANUFACT
977	MARC	MANUFACT

Autre exemple – Equivalent DSPPGMREF – 1/2

```
CREATE OR REPLACE FUNCTION pgmref(pgmparm VARCHAR(10), libparm VARCHAR(10))
   RETURNS TABLE(pgmname CHAR(10), objref CHAR(11), libref CHAR(11), objtype CHAR(10))
   LANGUAGE SQL
   READS SQL DATA
   BEGIN
      DECLARE cmd CHAR(300);
      DECLARE lg DECIMAL(15, 5);
      DECLARE EXIT HANDLER FOR SQLEXCEPTION SIGNAL SQLSTATE '99999'
                  SET MESSAGE_TEXT = 'Erreur d''appel de la procédure' ;
      SET CMD = 'QSYS/DSPPGMREF PGM(' CONCAT TRIM(libparm) CONCAT '/' CONCAT TRIM(pgmparm)
         CONCAT ') OUTPUT(*OUTFILE) OUTFILE(qtemp/dsppgm) ';
      SET lg = LENGTH(cmd);
      CALL QSYS.QCMDEXC(cmd, lg);
      RETURN
         SELECT WHPNAM, WHFNAM, WHLNAM, WHOTYP
            FROM qtemp.dsppgm ;
   END ;
```

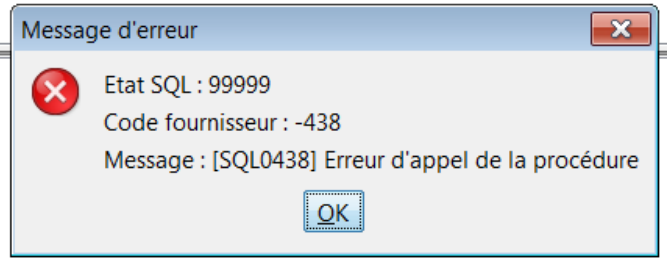
Autre exemple – Equivalent DSPPGMREF – 2/2

```
53 SELECT DISTINCT * FROM TABLE(pgmref('DECL_SIN_P', 'IBM_ASSUR1')) AS t1;
54
55
```

PGMNAME	OBJREF	LIBREF	OBJTYPE
DECL_SIN_P	ENQ_H_SIN	*LIBL	*PGM
DECL_SIN_P	QCMDEXC	*LIBL	*PGM
DECL_SIN_P	LAST_SIN_N	*LIBL	*DTAARA
DECL_SIN_P	DECL_SIN_D	IBM_ASSUR1	*FILE
DECL_SIN_P	CLIENTS	IBM_ASSUR1	*FILE
DECL_SIN_P	CT_DET_CTN	IBM_ASSUR1	*FILE

```
53 SELECT DISTINCT * FROM TABLE(pgmref('DECL_SIN_P', 'GLOPGLOP')) AS t1;
54
55
```

En cas d'erreur



Autre exemple – Audit d'utilisation

```
CREATE FUNCTION Adv_DeptEmployees(i_deptno VARCHAR(3))
RETURNS TABLE (emp_id CHAR(6), emp_name VARCHAR(35),
               educ_level SMALLINT, dept_name CHAR(20))
LANGUAGE SQL
DETERMINISTIC   NOT FENCED
EXTERNAL ACTION DISALLOW PARALLEL
CARDINALITY 10
BEGIN
 DECLARE AllCaps_DeptNum CHAR(3);
 SET AllCaps_DeptNum=UPPER(i_deptno);
 INSERT INTO audit_employee_access
            VALUES(USER, CURRENT TIMESTAMP);
 RETURN
  SELECT empno, lastname CONCAT ', ' CONCAT firstnme,
      edlevel, deptname
    FROM employee e INNER JOIN dept d
    ON e.workdept=AllCaps_DeptNum;
END;
```

UDTF SQL – Retour avec PIPE

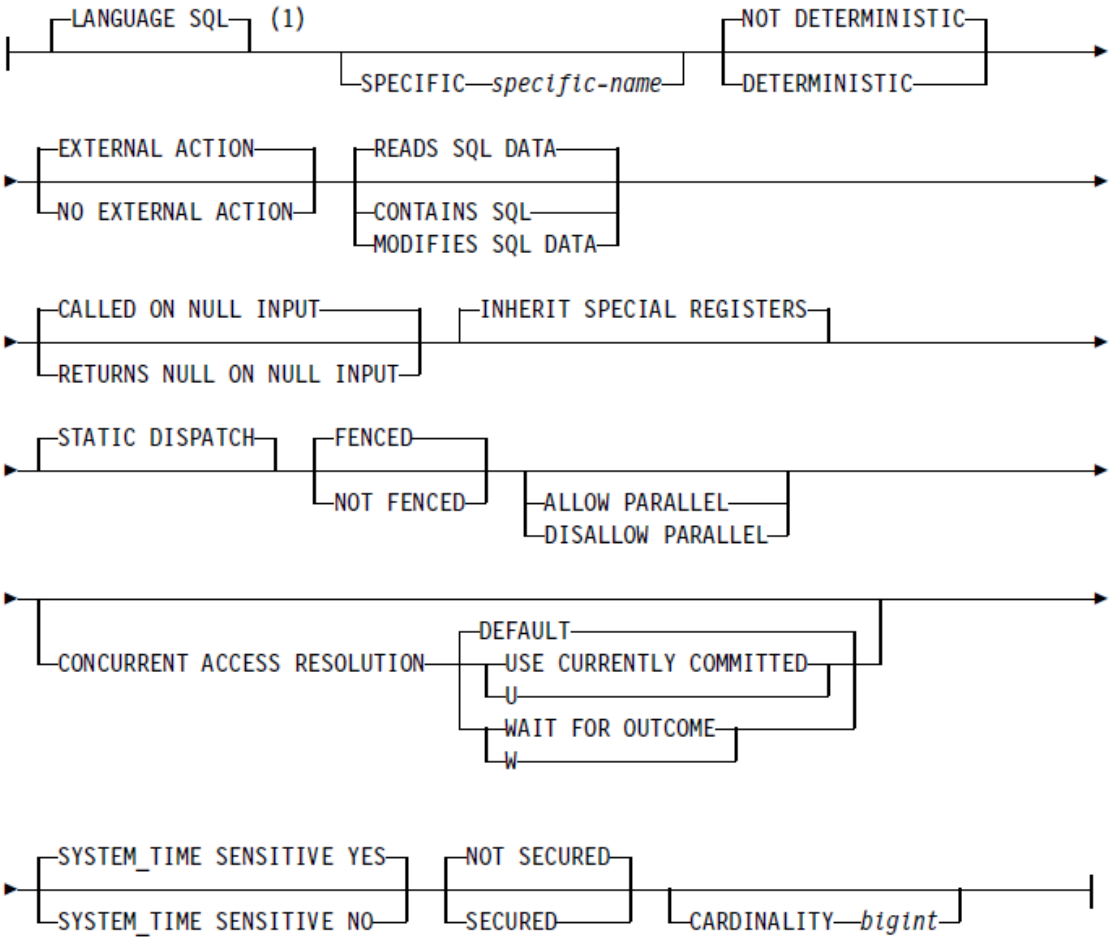
- PIPE : retourne une ligne à partir d'un curseur

```
875 CREATE OR REPLACE FUNCTION udtf2 (nosrv  SMALLINT)
876 RETURNS TABLE (srv SMALLINT, nom CHAR(12), nomsrv CHAR(10), libelsal CHAR(20))
877 LANGUAGE SQL READS SQL DATA
878 BEGIN
879  FOR c1 CURSOR FOR SELECT srv, nom, nomsrv, sal FROM tabserv JOIN tabempl USING(srv) WHERE srv=nosrv
880   DO
881    IF sal > 3000 THEN PIPE (srv, nom, nomsrv, 'Salaire > 3000 euros');
882    ELSE PIPE (srv, nom, nomsrv, 'Salaire < 3000 euros');
883    END IF;
884  END FOR;
885  RETURN;
886 END;
887
888 SELECT * FROM TABLE(udtf2(911)) AS srvemp;
889
```

SRV	NOM	NOMSRV	LIBELSAL
911	MICHEL	VENTES	Salaire < 3000 euros
911	ANNIE	VENTES	Salaire > 3000 euros
911	JACQUES	VENTES	Salaire < 3000 euros

Les options

option-list:



Les principales options

Option	Signification
CONTAINS SQL **READS SQL DATA** MODIFIES SQL DATA	Instructions SQL autorisées. Entre autres : - CONTAINS SQL : CALL - READS SQL DATA : SELECT et instructions de curseurs en lecture - MODIFIES SQL DATA : toute instruction SQL
NOT DETERMINISTIC DETERMINISTIC	DETERMINISTIC : pour une même valeur de paramètre en entrée la fonction retournera le même résultat en sortie. Utilisation d'un cache
EXTERNAL ACTION NO EXTERNAL ACTION	EXTERNAL ACTION : la procédure modifie l'état d'un objet non DB2 (dtaara…). Force l'appel de la fonction à chaque invocation
FENCED NOT FENCED	NOT FENCED : la fonction et l'instruction SQL qui l'invoque s'exécutent dans le même thread (plus performant)
CARDINALITY	Estimation du nombre de lignes retournées (pour optimisation)
SPECIFIC	Nom court = nom de l'objet de type *SRVPGM créé

3. Exploitation

Catalogue système

- La création d'UDTFs alimente les vues suivantes du catalogue :
  - SYSROUTINES
  - SYSFUNCS
  - SYSPARMS

```
79 SELECT * FROM QSYS2.SYSROUTINES WHERE ROUTINE_SCHEMA = 'AS425F'
80 AND ROUTINE_TYPE = 'FUNCTION' AND FUNCTION_TYPE = 'T';
81
82 SELECT * FROM QSYS2.SYSFUNCS  WHERE ROUTINE_SCHEMA = 'AS425F'
83 AND  FUNCTION_TYPE = 'T';
84
```

SPECIFIC_SCHEMA	SPECIFIC_NAME	ROUTINE_SCHEMA	ROUTINE_NAME	ROUTIN
AS425F	PGMREF	AS425F	PGMREF	2018-05
AS425F	UDTF1	AS425F	UDTF1	2018-03
AS425F	UDTF2	AS425F	UDTF2	2015-11

Visualisation du source d'une UDTF

- En utilisant le catalogue système (colonne ROUTINE_DEFINITION)

```
85 SELECT routine_definition
86    FROM qsys2.sysfuncs
87    WHERE routine_schema = 'AS425F' AND routine_name = 'PGMREF';
88
```

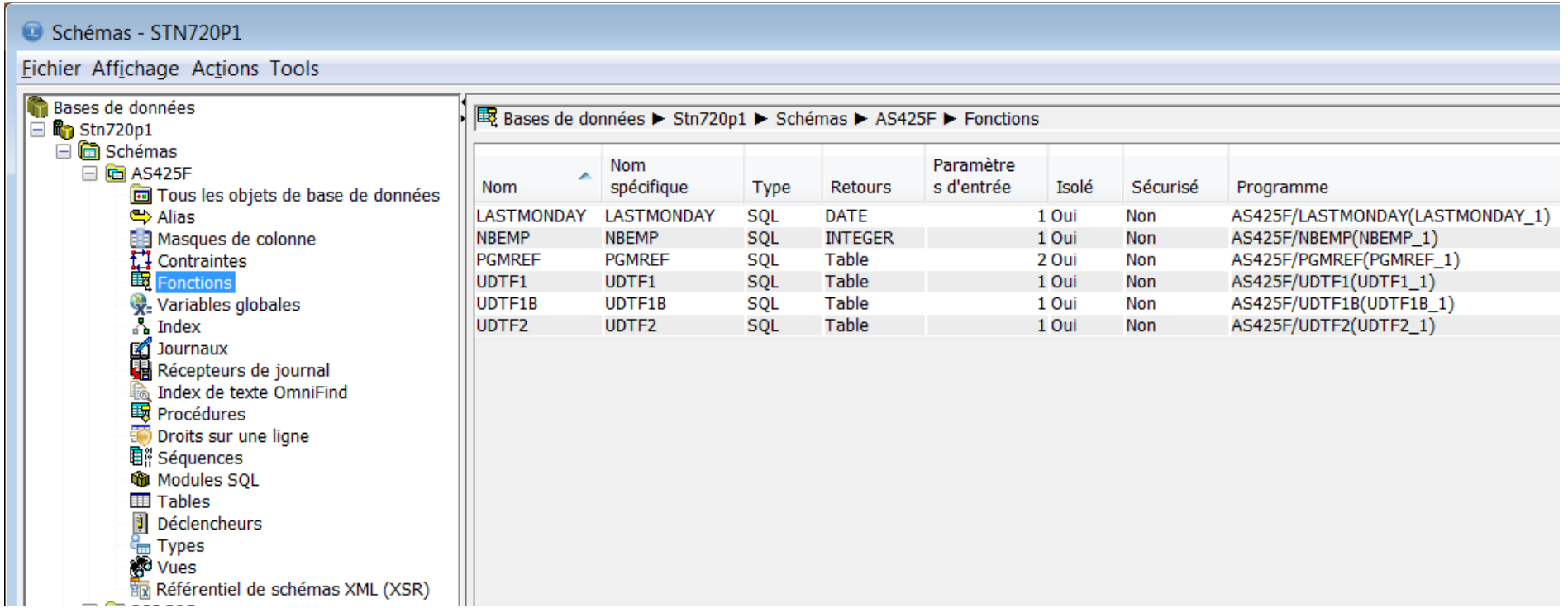
ROUTINE_DEFINITION
BEGINDECLARE CMD CHAR (300) ;DECLARE LG DECIMAL (15 , 5) ;DECLARE EXIT HANDLER FOR SQLEXCEPTION ...

Masquage du source d'une UDTF

```
91  SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
92
93  CALL CREATE_WRAPPED('CREATE OR REPLACE FUNCTION udtf1b (nosrv  SMALLINT)
94  RETURNS TABLE (srv SMALLINT, nom CHAR(12), nomsrv CHAR(10))
95  LANGUAGE SQL READS SQL DATA
96  RETURN
97   SELECT srv, nom, nomsrv FROM tabserv JOIN tabempl
98   USING(srv) WHERE srv=nosrv');
99
100 COMMIT;
101
102 SET TRANSACTION ISOLATION LEVEL NONE;
103
104 SELECT routine_definition
105    FROM qsys2.sysfuncs
106    WHERE routine_schema = 'AS425F' AND routine_name = 'UDTF1B';
```

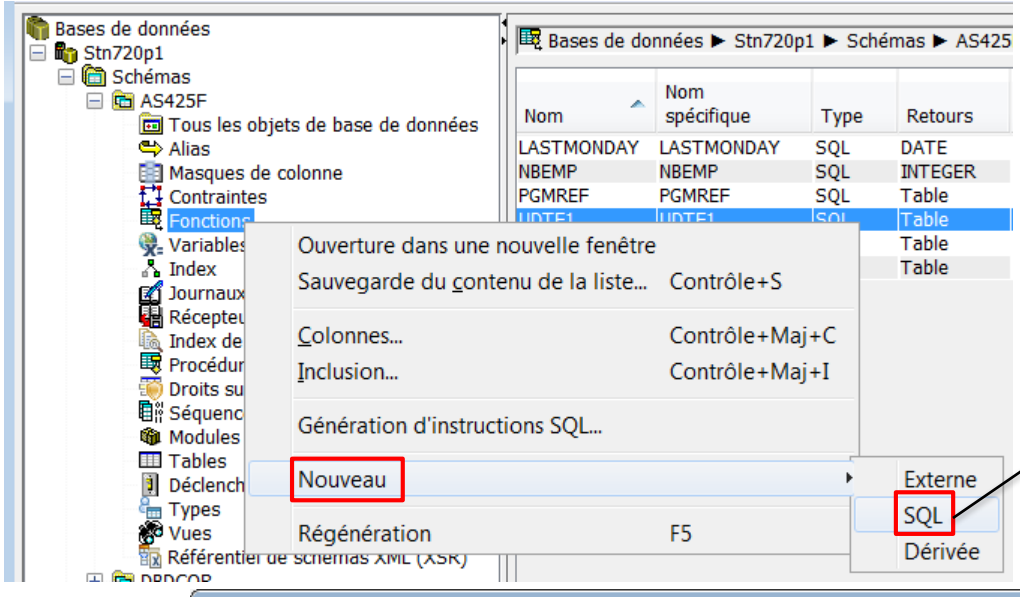
ROUTINE_DEFINITION
WRAPPED QSQ07020 aacxW8plW8FHG8pvG8FjG8Fz68Vv68pfl_pl4_ppWqpdW8pdW8pdX_pJz5u...

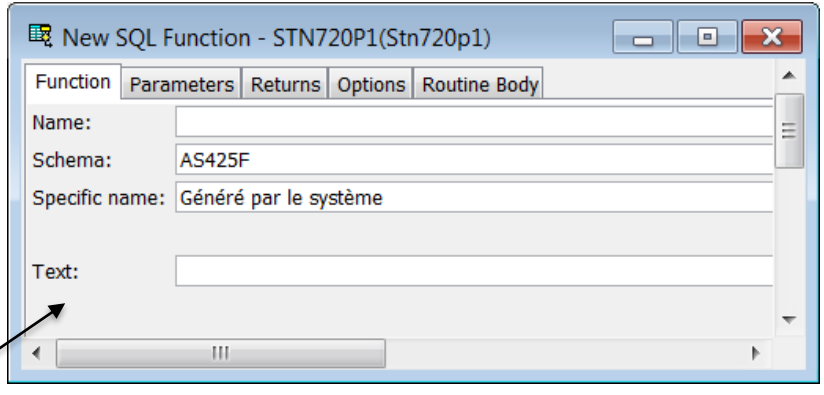
Exploitation par ACS ou IBM Navigator for i

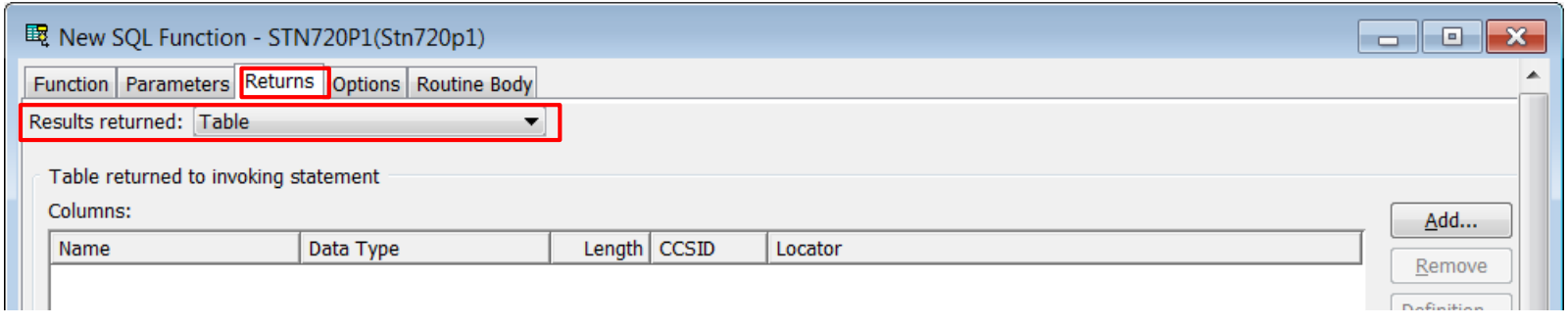


Nom	Nom spécifique	Type	Retours	Paramètres d'entrée	Isolé	Sécurisé	Programme
LASTMONDAY	LASTMONDAY	SQL	DATE	1	Oui	Non	AS425F/LASTMONDAY(LASTMONDAY_1)
NBEMP	NBEMP	SQL	INTEGER	1	Oui	Non	AS425F/NBEMP(NBEMP_1)
PGMREF	PGMREF	SQL	Table	2	Oui	Non	AS425F/PGMREF(PGMREF_1)
UDTF1	UDTF1	SQL	Table	1	Oui	Non	AS425F/UDTF1(UDTF1_1)
UDTF1B	UDTF1B	SQL	Table	1	Oui	Non	AS425F/UDTF1B(UDTF1B_1)
UDTF2	UDTF2	SQL	Table	1	Oui	Non	AS425F/UDTF2(UDTF2_1)

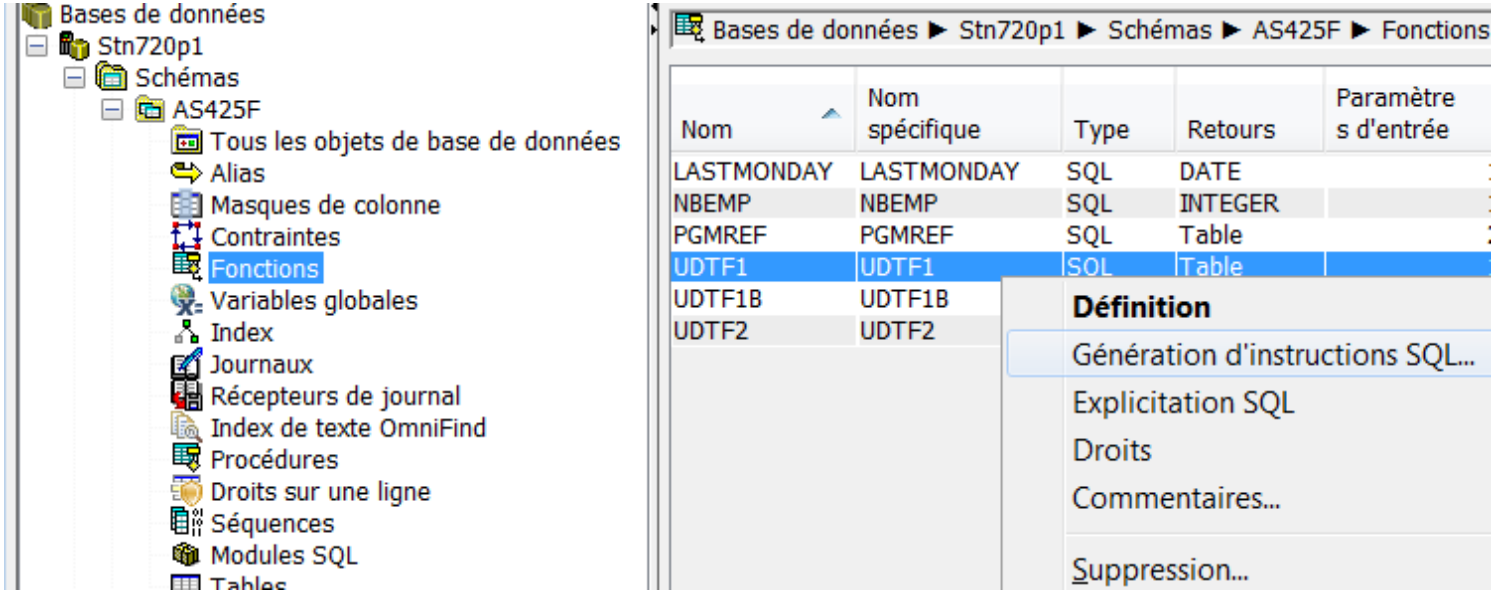
Exploitation par ACS ou IBM Navigator for i







Génération du source d'une UDTF – 1/2



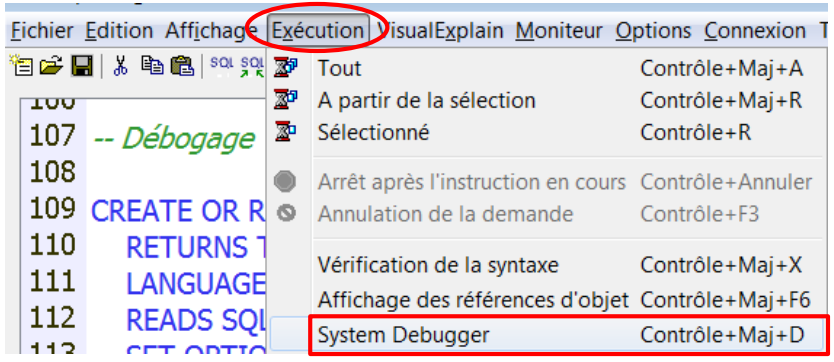
Génération du source d'une UDTF – 2/2

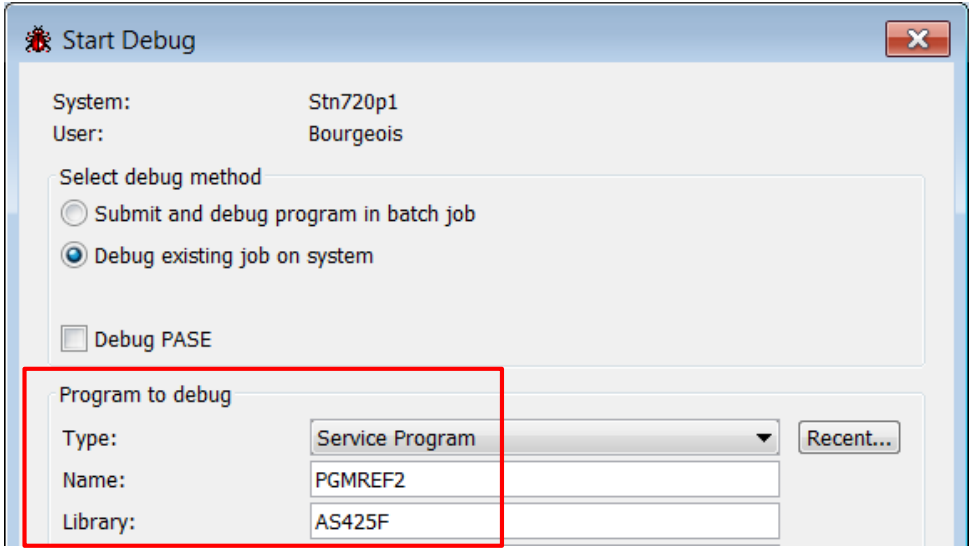
```
1  -- Générer SQL
2  -- Version :                  V7R2M0 140418
3  -- Générée le :               12/05/18 16:31:57
4  -- Base données relation :    STN720P1
5  -- Option normes :            DB2 for i
6  SET PATH "QSYS","QSYS2","SYSPROC","SYSIBMADM","AS425F" ;
7
8  CREATE FUNCTION AS425F.UDTF1 (
9          NOSRV SMALLINT)
10         RETURNS TABLE (
11         SRV SMALLINT ,
12         NOM CHAR(12) ,
13         NOMSRV CHAR(10) )
14         LANGUAGE SQL
15         SPECIFIC AS425F.UDTF1
16         NOT DETERMINISTIC
17         READS SQL DATA
18         CALLED ON NULL INPUT
19         SET OPTION  ALWBLK = *ALLREAD ,
20         ALWCPYDTA = *OPTIMIZE ,
21         COMMIT = *NONE ,
22         DECRESULT = (31, 31, 00) ,
23         DFTRDBCOL = AS425F ,
24         DYNDFTCOL = *NO ,
25         DYNUSRPRF = *USER ,
26         SRTSEQ = *HEX
27         RETURN
28 SELECT SRV , NOM , NOMSRV FROM TABSERV JOIN TABEMPL
29 USING ( SRV ) WHERE SRV = NOSRV  ;
30
31 GRANT ALTER , EXECUTE
32 ON SPECIFIC FUNCTION AS425F.UDTF1
33 TO BOURGEOIS WITH GRANT OPTION ;
```

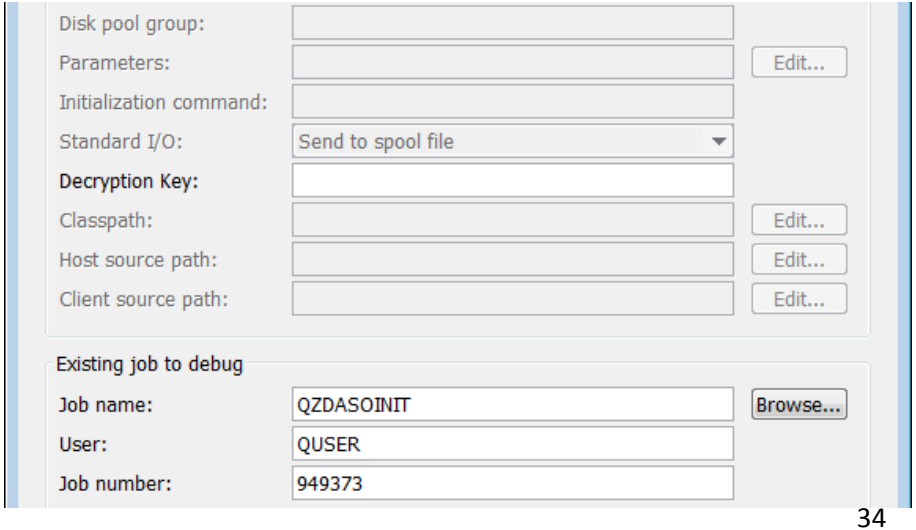
Débogage d'une UDTF

```
CREATE OR REPLACE FUNCTION pgmref2(pgmparm VARCHAR(10), libparm VARCHAR(10))
   RETURNS TABLE(pgmname CHAR(10), objref CHAR(11), libref CHAR(11), objtype CHAR(10))
   LANGUAGE SQL
   READS SQL DATA
   SET OPTION DBGVIEW = *SOURCE
   BEGIN
      DECLARE cmd CHAR(300);
      DECLARE lg DECIMAL(15, 5);
      DECLARE EXIT HANDLER FOR SQLEXCEPTION SIGNAL SQLSTATE '99999'
                  SET MESSAGE_TEXT = 'Erreur d''appel de la procédure' ;
      SET CMD = 'QSYS/DSPPGMREF PGM(' CONCAT TRIM(libparm) CONCAT '/' CONCAT TRIM(pgmparm)
         CONCAT ') OUTPUT(*OUTFILE) OUTFILE(qtemp/dsppgm) ';
      SET lg = LENGTH(cmd);
      CALL QSYS.QCMDEXC(cmd, lg);
      RETURN
         SELECT WHPNAM, WHFNAM, WHLNAM, WHOTYP
            FROM qtemp.dsppgm ;
   END ;
```

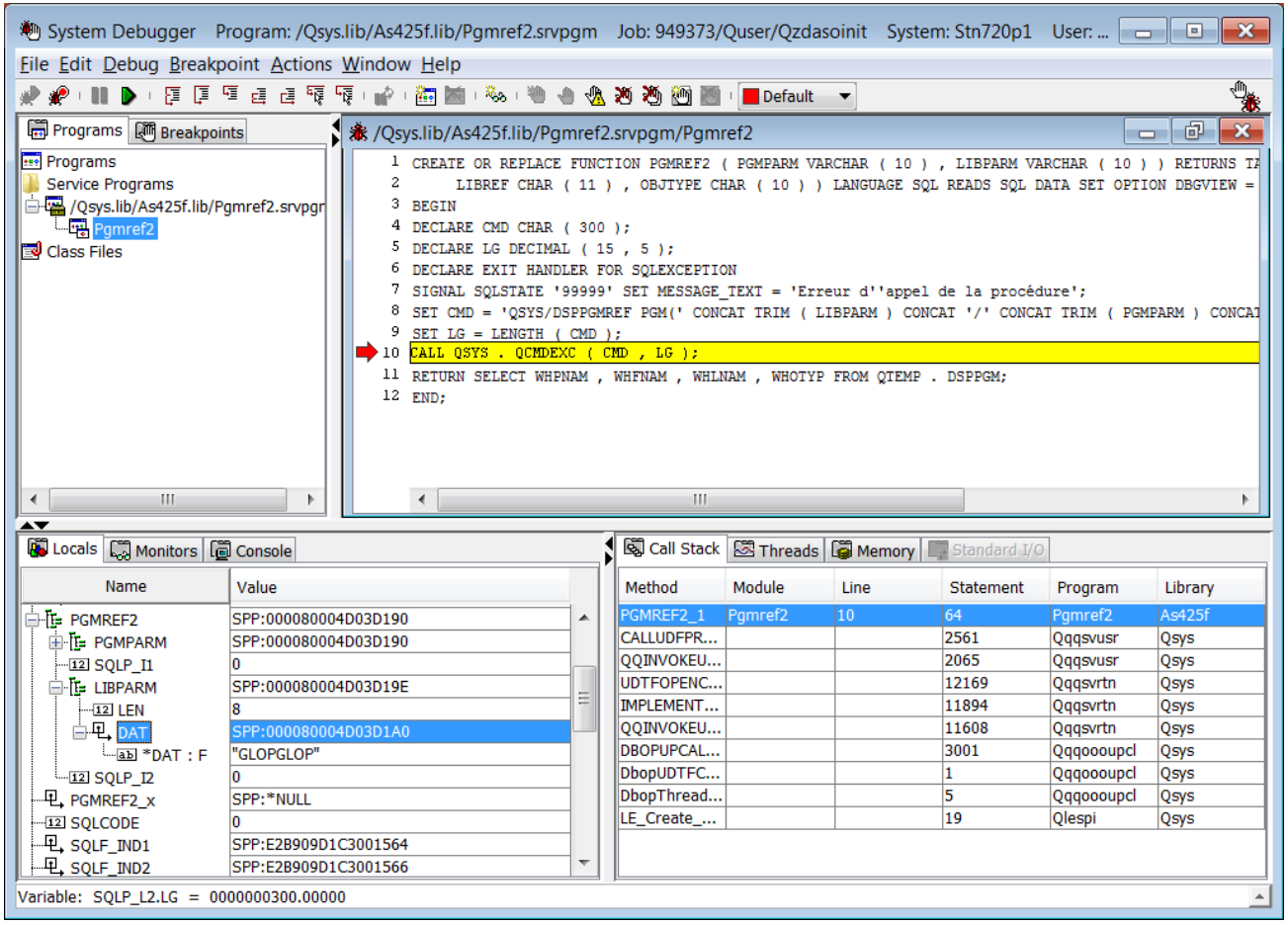
Débogage d'une UDTF – System Debugger



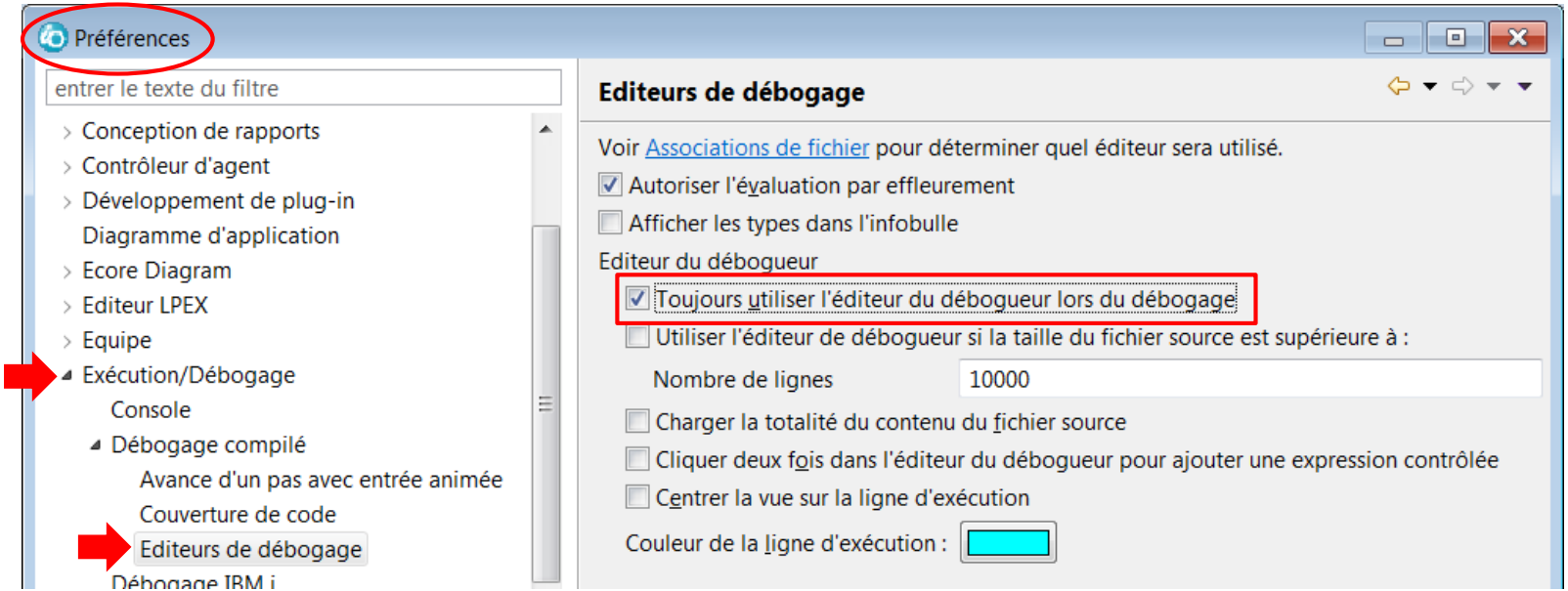




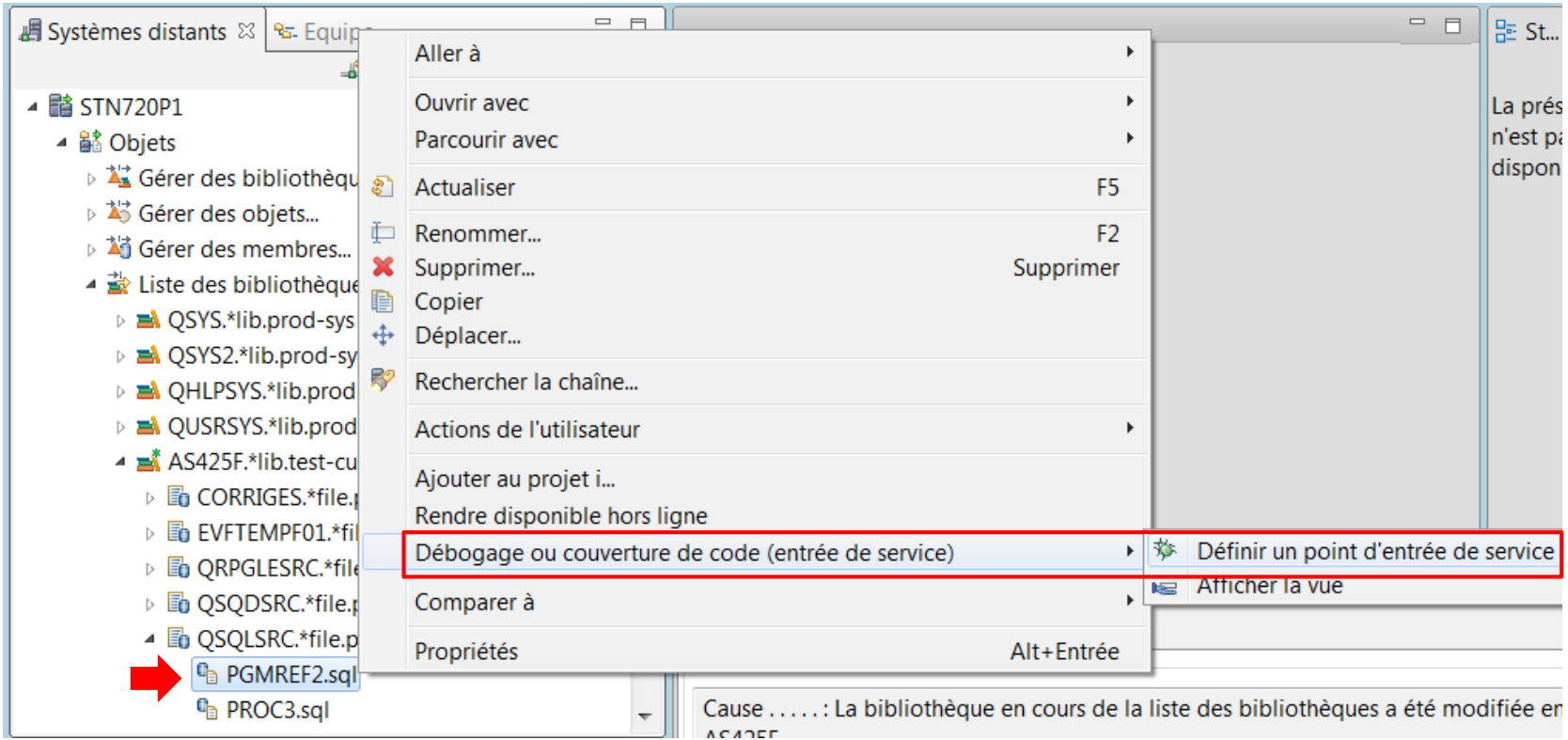
Débogage d'une UDTF – System Debugger



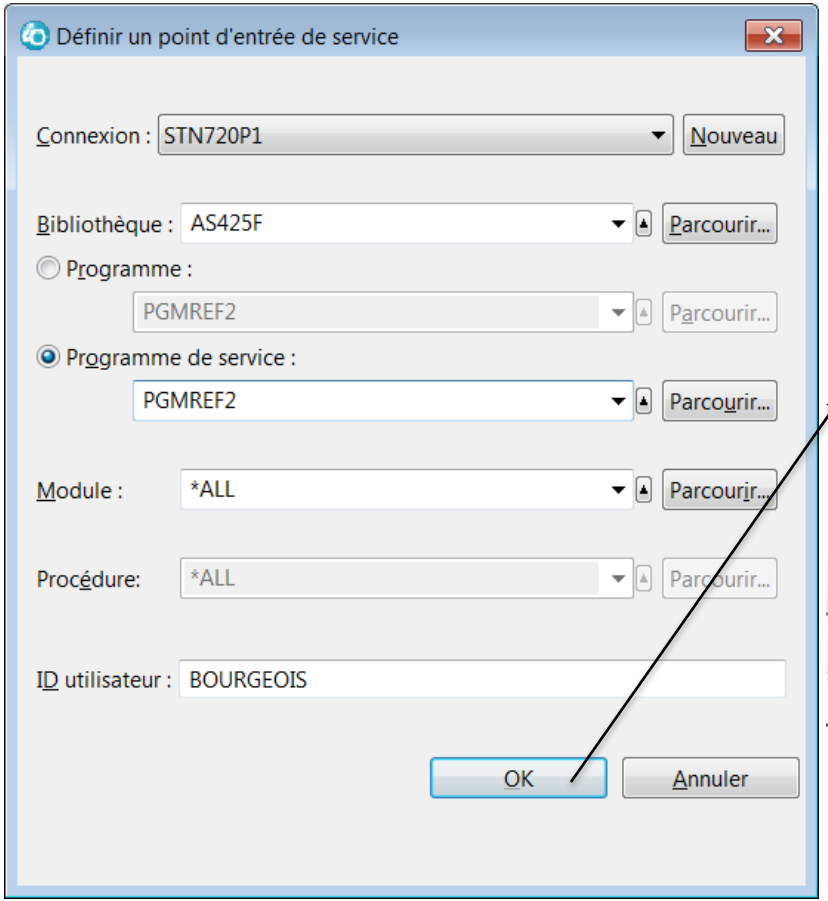
Débogage d'une UDTF – RDi

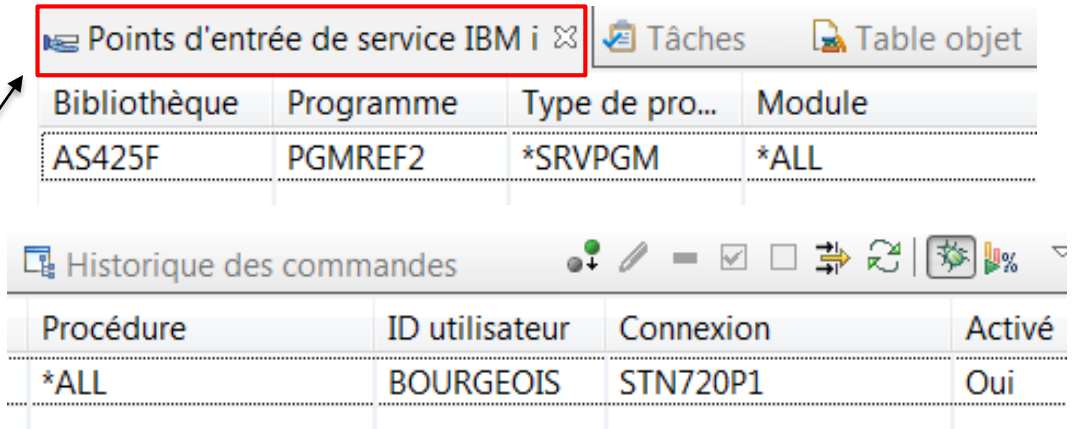


Débogage d'une UDTF – RDi



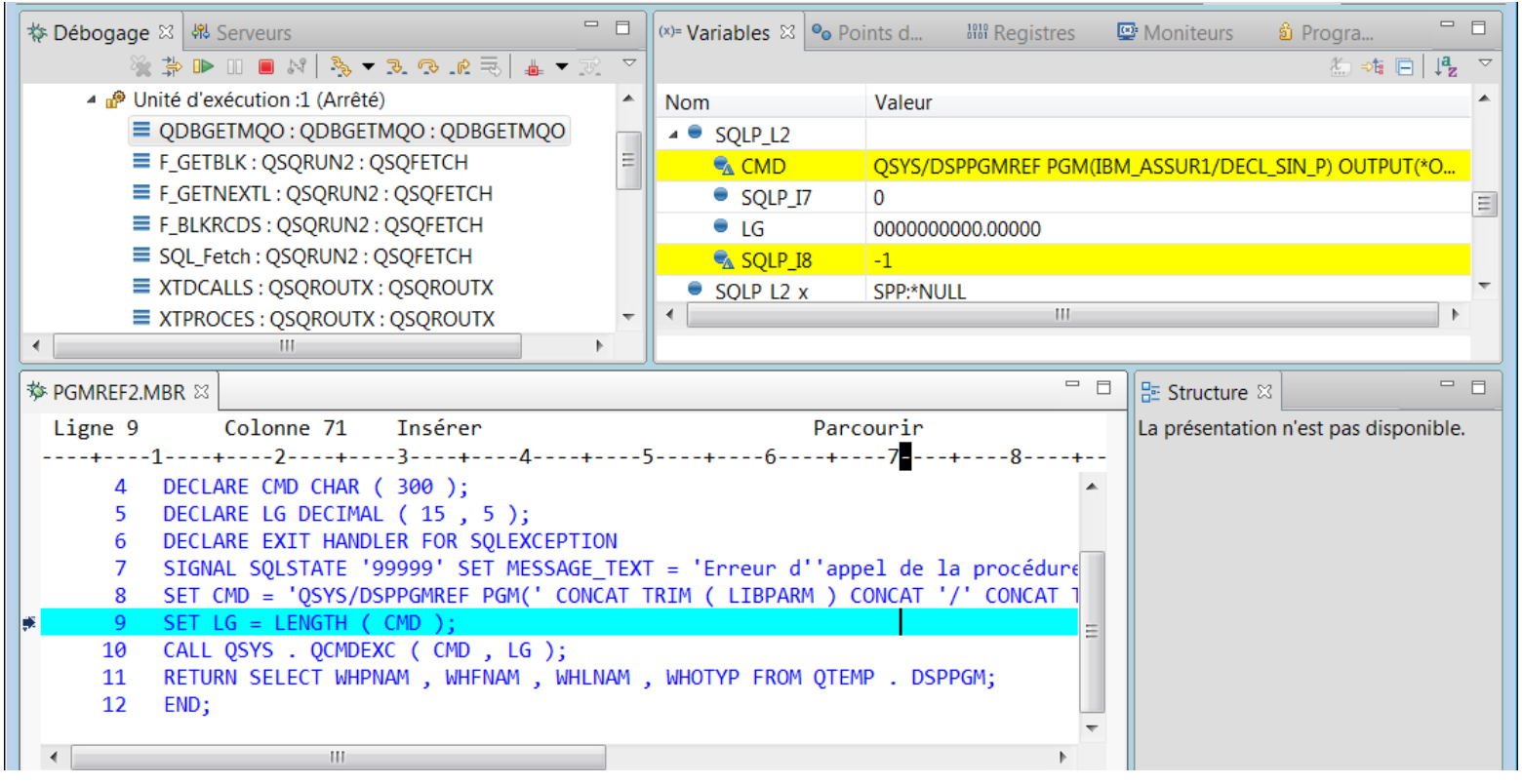
Débogage d'une UDTF – RDi





Débogage d'une UDTF – RDi

```
SELECT DISTINCT * FROM TABLE(pgmref2('DECL_SIN_P', 'IBM_ASSUR1')) AS t1;
```



Signature

- Lors de la création d'une fonction, une signature est générée à partir de 2 éléments :
  - Le nom long de la fonction
  - Le type de chaque paramètre
- Il n'est pas possible de créer dans le même schéma 2 procédures avec la même signature
  - Mais c'est possible avec deux fonctions qui portent le même nom et qui ont des paramètres de type différent !
  - Ces 2 fonctions ont des signatures uniques, elles peuvent résider dans le même schéma :
    - `CREATE FUNCTION f1 (p1 INT, p2 CHAR(5))`
    - `CREATE FUNCTION f1 (p1 INT, p2 CHAR(5))`
  - Ces 2 fonctions ont des signatures dupliquées, elles ne peuvent pas résider dans le même schéma :
    - `CREATE FUNCTION f1 (p1 INT, p2 CHAR(5))`
    - `CREATE FUNCTION f1 (p1 INT, parm2 CHAR(100))`

Résolution des appels de fonction

- Plusieurs fonctions pouvant porter le même nom, DB2 déterminera celle à appeler à l'aide :
  - De la signature
  - Du chemin (PATH) :
    - Si NAMING(*SYS) : *LIBL
    - Si NAMING(*SQL) : QSYS QSYS2 SYSPROC SYSIBMADM current_schema

- La correspondance exacte des types est préférée, mais une hiérarchie de promotion existe ; par exemple :
  - CHAR pourra utiliser VARCHAR ou CLOB
  - INTEGER pourra utiliser SMALLINT, DECIMAL…
  - Les constantes sont traitées en VARCHAR

Détails dans le workshop "SQL avancé"

4. UDTFs fournies par IBM

Différentes catégories des UDTFs DB2 for i

- Les UDTFs de la famille DB2

- Les UDTFs spécifiques à DB2 for i
  - Services DB2 for i
  - Services IBM i

- Les UDTFs fournies dans la bibliothèque SYSTOOLS
  - Fonctions SQL HTTP

Les UDTFs de la famille DB2

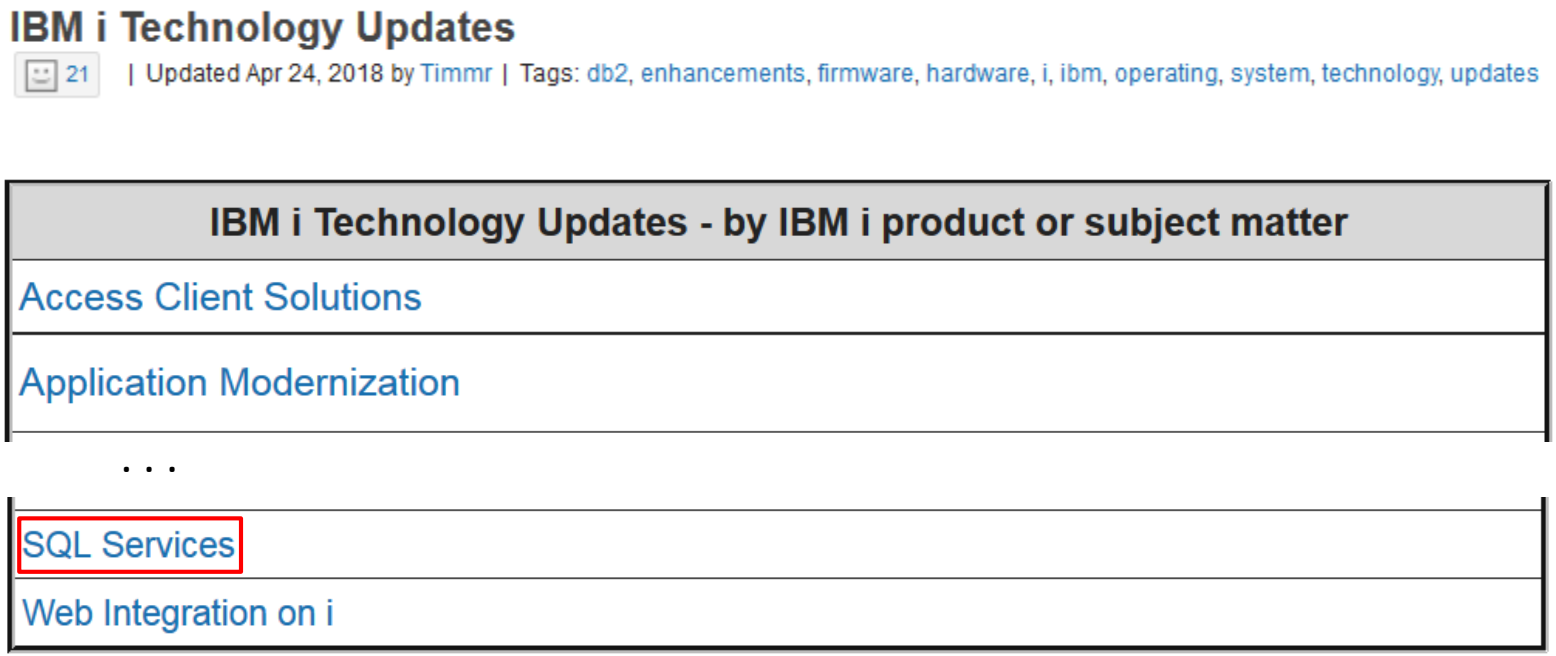
	Description	Db2 for z/OS	Db2 for i	Db2 (for LUW)
BASE_TABLE	Returns the object names and schema names of the object found for an alias	X	X	X
JSON_TABLE	Returns a result table from the evaluation of SQL/JSON path expressions	partial	X	-
MQREADALL	Returns a table of messages and message metadata from MQSeries	X	X	X
MQREADALLCLOB	Returns a table of messages and message metadata from MQSeries	X	X	X
MQRECEIVEALL	Returns a table of messages and message metadata from MQSeries with removal of the messages	X	X	X
MQRECEIVEALLCLOB	Returns a table of messages and message metadata from MQSeries with removal of the messages	X	X	X
XMLTABLE	Returns a result table from the evaluation of XPath expressions	X	X	partial

Les services IBM i et DB2 for i

- Les services IBM i et DB2 for i
  - Ce sont des vues, procédures et fonctions **SQL** fournies en standard permettant d'accéder à des fonctions IBM i **système**
  - Objectif : utiliser le langage SQL pour accéder à des fonctions système IBM i
  - Une alternative aux commandes CL et aux APIs
  - On parle également de **SQL as a Service**
  - Pour qui ?
    - Les administrateurs système
    - Les administrateurs de la sécurité
    - Les administrateurs de la base de données
    - Mais également les développeurs
  - Pourquoi ?
    - Standard, compétences, puissance, accès externe

Les services IBM i et DB2 for i

- Ces services sont documentés sur le site **IBM i Technology Updates**



IBM i Technology Updates

21 | Updated Apr 24, 2018 by Timmr | Tags: db2, enhancements, firmware, hardware, i, ibm, operating, system, technology, updates

IBM i Technology Updates - by IBM i product or subject matter
Access Client Solutions
Application Modernization
. . .
SQL Services
Web Integration on i

Les services IBM i et DB2 for i

- Ces services sont documentés sur le site **IBM i Technology Updates**

IBM i Service	Type of Service	IBM i 7.3	IBM i 7.2	IBM i 7.1
PTF Services				
QSYS2.GROUP_PTF_INFO	View	Base	Base	SF99701 Level 6
QSYS2.PTF_INFO	View	Enhanced in Base	Base	SF99701 Level 23
SYSTOOLS.GROUP_PTF_CURRENCY	View	Base Enhanced: SF99703 Level 3	SF99702 Level 3 Enhanced: SF99702 Level 14	SF99701 Level 32 Enhanced: SF99701 Level 41
. . .				
SYSPROC.SET_COLUMN_ATTRIBUTE()	Procedure	Base	Base	Base
Message Handling Services				
QSYS2.HISTORY_LOG_INFO()	UDTF	SF99703 Level 3 Enhanced: SF99703 Level 7	SF99702 Level 14 Enhanced: SF99702 Level 19	-
QSYS2.JOBLOG_INFO	UDTF	Enhanced in Base	SF99702 Level 3	SF99701 Level 32

Les services IBM i et DB2 for i – Les UDTFs

- Gestion des **messages**
  - QSYS2.HISTORY_LOG_INFO()
  - QSYS2.JOBLOG_INFO
- Gestion des **objets**
  - QSYS2.OBJECT_STATISTICS()
- Gestion des **travaux**
  - QSYS2.ACTIVE_JOB_INFO()
  - QSYS2.GET_JOB_INFO()
  - QSYS2.JOB_INFO()
  - QSYS2.MEMORY_POOL()
  - QSYS2.SYSTEM_STATUS()
- Gestion des **spoules**
  - QSYS2.OUTPUT_QUEUE_ENTRIES()
- Gestion des **journaux**
  - QSYS2.DISPLAY_JOURNAL()
- Gestion des **applications**
  - QSYS2.STACK_INFO()
  - QSYS2.PARSE_STATEMENT()

Exemple de service sous forme d'UDTF

- Historique de travail : service JOB_LOG_INFO

La dernière commande CL exécutée par un travail donné

```
166 SELECT MESSAGE_TEXT, MESSAGE_TIMESTAMP
167    FROM TABLE(QSYS2.JOBLOG_INFO('211302/BOURGEOIS/QPADEV0008')) A
168    WHERE A.MESSAGE_TYPE = 'REQUEST'
169    ORDER BY ORDINAL_POSITION DESC
170    FETCH FIRST 1 ROW ONLY;
171
```

MESSAGE_TEXT	MESSAGE_TIMESTAMP
CHGSYSVAL SYSVAL(QPWDRULES) VALUE(*PWDSYSVAL)	2017-05-04 18:54:41.267109

Exemple de service sous forme d'UDTF

- Liste d'objets : service OBJECT_STATISTICS;

```
143 SELECT *
144   FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*ALL')) AS t;
```

Tous les objets de la bib AS425F

OBJNAME	OBJTYPE	OBJOWNER	OBJDEFINER	OBJCREATED	OBJSIZE	OBJTEXT
AUTORENS	*PGM	BOURGEOIS	BOURGEOIS	2017-05-04 19:07:18.000000	131072	Pour alimenter champs auto-renseignés
BLOBRPG1	*PGM	BOURGEOIS	BOURGEOIS	2014-10-06 10:03:15.000000	135168	Exemple 1 chapitre 14
BLOBRPG2	*PGM	BOURGEOIS	BOURGEOIS	2014-10-06 10:25:30.000000	262144	Exemple 3 chapitre 14
BLOBRPG3	*PGM	BOURGEOIS	BOURGEOIS	2014-10-06 10:45:22.000000	139264	Exemple 4 chapitre 14
BLOBRPG4	*PGM	BOURGEOIS	BOURGEOIS	2014-10-06 12:36:50.000000	143360	Exemple 5 chapitre 14
BLOBRPG5	*PGM	BOURGEOIS	BOURGEOIS	2014-10-06 12:48:29.000000	126976	Exemple 6 chapitre 14
EMP_DETAIL	*PGM	BOURGEOIS	BOURGEOIS	2017-02-28 11:56:02.000000	118784	Détail employé

56 attributs récupérables

```
146 SELECT *
147   FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*JRN *JRNRCV', '*ALLSIMPLE')) AS t;
```

OBJNAME	OBJTYPE	OBJOWNER	OBJDEFINER	OBJCREATED	OBJSIZE	OBJTEXT	OBJLO
JRN1	*JRN	-	-	-			
QSQJRN	*JRN	-	-	-			
QSQJRN1125	*JRNRCV	-	-	-			
RCV1078	*JRNRCV	-	-	-			

Les journaux et récepteurs de journaux de la bib AS425F

Exemple de service sous forme d'UDTF

- Liste d'objets : service OBJECT_STATISTICS;

```
149 SELECT SUM(OBJSIZE)
150    FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*FILE')) AS t
151    WHERE OBJATTRIBUTE = 'LF';
```

00001
3837952

La taille de tous les index de la bib AS425F

La valeur d'audit des tables SQL de la bib AS425F

```
153 SELECT OBJNAME, OBJTYPE, SQL_OBJECT_TYPE, OBJOWNER, OBJECT_AUDIT
154    FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*FILE')) AS t
155    WHERE SQL_OBJECT_TYPE = 'TABLE';
```

OBJNAME	OBJTYPE	SQL_OBJECT_TYPE	OBJOWNER	OBJECT_AUDIT
MKTGSALES	*FILE	TABLE	BOURGEOIS	*NONE
MQTSALSRV	*FILE	TABLE	BOURGEOIS	*NONE
PERFS	*FILE	TABLE	QDFTOWN	*NONE
QZG0000174	*FILE	TABLE	BOURGEOIS	*NONE
SALARIES	*FILE	TABLE	BOURGEOIS	*CHANGE
SERVICES	*FILE	TABLE	QDFTOWN	*NONE

Exemple de service sous forme d'UDTF

- Liste d'objets : service OBJECT_STATISTICS;

```
160 SELECT OBJNAME, OBJTYPE, JOURNAL_LIBRARY, JOURNAL_NAME, JOURNAL_IMAGES, OMIT_JOURNAL_ENTRY
161    FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*FILE')) AS t
162    WHERE JOURNALED = 'YES';
163
```

Les informations de journalisation

OBJNAME	OBJTYPE	JOURNAL_LIBRARY	JOURNAL_NAME	JOURNAL_IMAGES	OMIT_JOURNAL_ENTRY
EMPLOYES	*FILE	AS425F	JRN1	*AFTER	*NONE
IDX_UP_NOM	*FILE	AS425F	JRN1	*AFTER	*NONE
MKTGSALES	*FILE	AS425F	QSQJRN	*BOTH	*OPNCLO
SERV	*FILE	AS425F	JRN1	*AFTER	*NONE

```
165 SELECT objname, objtype, change_timestamp, last_used_timestamp, days_used_count
166    FROM TABLE(QSYS2.OBJECT_STATISTICS('AS425F', '*ALL')) as t
167    ORDER BY change_timestamp DESC;
168
```

Les informations de modification et d'utilisation

OBJNAME	OBJTYPE	CHANGE_TIMESTAMP	LAST_USED_TIMESTAMP	DAYS_USED_COUNT
EMP_DETAIL	*PGM	2017-02-28 11:56:02.000000	2017-03-01 00:00:00.000000	2
QRPGLESRC	*FILE	2017-02-28 11:43:09.000000	2017-02-28 00:00:00.000000	21
SOURCES	*FILE	2016-11-18 09:28:57.000000	2017-05-04 00:00:00.000000	20
TABEMPL4	*FILE	2016-08-25 15:12:22.000000	2016-12-16 00:00:00.000000	3

Exemple de service sous forme d'UDTF

- Travaux actifs : service ACTIVE_JOB_INFO

```
221 SELECT *
222    FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) AS T;
223
```

Liste des travaux actifs

ORDINAL_POSITION	JOB_NAME	INTERNAL_JOB_ID	SUBSYSTEM	S
1	214823/QSYS/QBATCH	0051000100046B009F1ED23A53302001	QBATCH	Q
2	214824/QSYS/QCMN	00510001000477009F1ED23A5D01D001	QCMN	Q
3	214855/QUSER/QACSOTP	00510001000647009F1ED23AC10EF001	QCMN	Q
4	214867/QUSER/QLZPSERV	00510001000677009F1ED23AF3DE9001	QCMN	Q
5	214844/QUSER/QNMAPINGD	00510001000[illegible]9F1ED23AAF8D8001	QCMN	Q

```
224 SELECT COUNT(*)
225    FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) AS T;
226
```

00001
316

Nombre de travaux actifs

Exemple de service sous forme d'UDTF

- Combinaison des services ACTIVE_JOB_INFO et USER_INFO

```
227 SELECT COUNT(*)
228    FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) AS T1
229    JOIN QSYS2.USER_INFO AS T2 USING(AUTHORIZATION_NAME)
230    WHERE SPECIAL_AUTHORITIES LIKE '%ALLOBJ%';
```

00001
189

Combien de travaux tournent avec un profil *ALLOBJ

```
232 SELECT COUNT(*)
233    FROM TABLE(QSYS2.ACTIVE_JOB_INFO()) AS T1
234    JOIN QSYS2.USER_INFO AS T2 USING(AUTHORIZATION_NAME)
235    WHERE SPECIAL_AUTHORITIES LIKE '%ALLOBJ%' AND NO_PASSWORD_INDICATOR = 'NO';
```

00001
80

Combien de travaux tournent avec un profil *ALLOBJ et un mot de passe

Exemple de service sous forme d'UDTF

- Travaux actifs : service ACTIVE_JOB_INFO
  - C'est une UDTF qui peut prendre 4 paramètres :
    - RESET_STATISTIC (NO / YES)
      - Pour réinitialiser les mesures de temps écoulés
    - SUBSYSTEM_LIST_FILTER
      - Pour filtrer sur un ou plusieurs sous-systèmes
    - JOB_NAME_FILTER
      - Pour filtrer sur un ou plusieurs travaux
    - CURRENT_USER_LIST_FILTER
      - Pour filtrer sur un ou plusieurs profils

```
237 SELECT *
238    FROM TABLE(QSYS2.ACTIVE_JOB_INFO('NO', NULL, NULL, 'QPGMR')) AS T;
239
```

Les travaux actifs sous QPGMR

ORDINAL_POSITION	JOB_NAME	SUBSYSTEM	AUTHORIZATION_NAME	JOB_TYPE
1	214821/QPGMR/QSYSSCD	QCTL	QPGMR	BCH
2	214841/QPGMR/QSERVER	QSERVER	QPGMR	ASJ
3	215024/QPGMR/QZLSSERVER	QSERVER	QPGMR	BCH

Exemple de service sous forme d'UDTF

- Travaux actifs : service ACTIVE_JOB_INFO

```
241 SELECT JOB_NAME, AUTHORIZATION_NAME,
242        ELAPSED_CPU_PERCENTAGE,
243        ELAPSED_TOTAL_DISK_IO_COUNT,
244        ELAPSED_PAGE_FAULT_COUNT, T.*
245   FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_NAME_FILTER => 'QZDASOINIT',
246                                    SUBSYSTEM_LIST_FILTER =>'QUSRWRK')) AS T
247   ORDER BY ELAPSED_CPU_PERCENTAGE DESC LIMIT 10;
```

SELECT JOB_NAME, AUTHORIZATION_NAME, ELAPSED_CPU_PERCENTAGE,ELAPSED_TOTAL_DISK_IO_COUNT, ... - Lp24ut27.rch.stglabs.ibm.com(Lp24ut27)

JOB_NAME	AUTHORIZATION_NAME	ELAPSED_CPU_PERCENTAGE	ELAPSED_TOTAL_DISK_IO_COUNT	ELAPSED_PAGE_FAULT_COUNT
566453/QUSER/QZDASOINIT	SCOTTF	94.9	206426	3718
554692/QUSER/QZDASOINIT	KADLER	0.0	0	0
564478/QUSER/QZDASOINIT	SCOTTF	0.0	950	893
564479/QUSER/QZDASOINIT	SCOTTF	0.0	0	0
566454/QUSER/QZDASOINIT	SCOTTF	0.0	59	21
566494/QUSER/QZDASOINIT	SCOTTF	0.0	574	183
566495/QUSER/QZDASOINIT	SCOTTF	0.0	1545	1047
566694/OUSER/OZDASOINIT	OUSER	0.0	0	0

Les 10 travaux QZDASOINIT qui consomment le plus de CPU

Exemple de service sous forme d'UDTF

- Liste de travaux : service JOB_INFO
  - C'est une UDTF qui peut prendre 5 paramètres :
    - JOB_STATUS_FILTER
    - JOB_TYPE_FILTER
    - JOB_SUBSYSTEM_FILTER
    - JOB_USER_FILTER
    - JOB_SUBMITTER_FILTER

```
261 SELECT *
262    FROM TABLE(QSYS2.JOB_INFO(JOB_STATUS_FILTER => '*JOBQ')) AS T;
263
```

Les travaux en JOBQ

JOB_NAME	JOB_STATUS	JOB_TYPE	JOB_SUBSYSTEM	JOB_INFORMATION
226657/BOURGEOIS/QDFTJOBD	JOBQ	BCH	QBATCH	YES

Les travaux de QHTTPSVR

```
262 SELECT *
263    FROM TABLE(QSYS2.JOB_INFO(JOB_SUBSYSTEM_FILTER => 'QHTTPSVR',
264                              JOB_USER_FILTER => '*ALL')) AS T
265    ORDER BY JOB_ENTERED_SYSTEM_TIME;
266
```

JOB_NAME	JOB_INFORMATION	JOB_STATUS	JOB_TYPE
214928/QTMHHTTP/ADMIN	YES	ACTIVE	BCH
214931/QTMHHTTP/PHLADMIN	YES	ACTIVE	BCH

Exemple de service sous forme d'UDTF

- Combinaison des services ACTIVE_JOB_INFO et GET_JOB_INFO

```
248 WITH ACTIVE_USER_JOBS(JOB_NAME, CPU_TIME, RUN_PRIORITY)
249    AS (SELECT JOB_NAME, CPU_TIME, RUN_PRIORITY
250          FROM TABLE(ACTIVE_JOB_INFO()) AS T1
251          WHERE JOB_TYPE <> 'SYS')
252    SELECT JOB_NAME, CPU_TIME, RUN_PRIORITY, V_SQL_STATEMENT_TEXT, ABS(CURRENT TIMESTAMP -
253       V_SQL_STMT_START_TIMESTAMP) AS SQL_STMT_DURATION FROM ACTIVE_USER_JOBS,
254    TABLE(QSYS2.GET_JOB_INFO(JOB_NAME)) AS T2
255    WHERE V_SQL_STMT_STATUS = 'ACTIVE' ORDER BY SQL_STMT_DURATION DESC;
256
```

Q_JOB_NAME	CPU_TIME	RUN_PRIORITY	V_SQL_STATEMENT_TEXT	SQL_STMT_DURATION
562630/QUSER/QRWTSRVR	7802	20	DECLARE C2 CURSOR FOR S2	53.102170
568166/SCOTTF/QPADEV000T	8847	20	INSERT into toystore.sales_2 ...	27.158785
566454/QUSER/QZDASOINIT	763	20	WITH ACTIVE_USER_JOBS (Q_JOB_NAME, CPU...	0.000078

Les travaux actifs avec les requêtes SQL les plus longues

Exemple de service sous forme d'UDTF

- Postes d'un journal : service DISPLAY_JOURNAL

Qui a supprimé des lignes dans la table EMPLOYES ces 7 derniers jours ?

```
309 SELECT ENTRY_DATA, ENTRY_TIMESTAMP, JOURNAL_ENTRY_TYPE, COUNT_OR_RRN AS ROW_DELETED,
310    "CURRENT_USER", RTRIM(JOB_NUMBER) CONCAT '/' CONCAT RTRIM(JOB_USER) CONCAT '/' CONCAT
311    RTRIM(JOB_NAME) AS Qualified_Job_Name
312    FROM TABLE(QSYS2.DISPLAY_JOURNAL('PBSQL', 'QSQJRN',
313                                     STARTING_TIMESTAMP => CURRENT TIMESTAMP - 7 DAYS,
314                                     JOURNAL_ENTRY_TYPES => 'DL',
315                                     OBJECT_LIBRARY => 'PBSQL',
316                                     OBJECT_NAME => 'EMPLOYES',
317                                     OBJECT_OBJTYPE => '*FILE',
318                                     OBJECT_MEMBER => 'EMPLOYES')) AS T
319    ORDER BY ENTRY_TIMESTAMP DESC;
320
```

ENTRY_DATA	ENTRY_TIMESTAMP	JOURNAL_ENTRY_TYPE	ROW_DELETED	CURRENT_USER	QUALIFIED_JOB_NAME
005AC8C5D3C...	2017-05-14 17:43:...	DL	25	BOURGEOIS	226922/QUSER/QZDASOINIT
0078D1C5C1D...	2017-05-14 17:43:...	DL	3	BOURGEOIS	226922/QUSER/QZDASOINIT

Exemple de service sous forme d'UDTF

- "Objets" SQL utilisés par une instruction SQL : service PARSE_STATEMENT

Les "objets" SQL utilisés par une requête SQL (tables, vues, index… colonnes, curseurs…)

```
459 SELECT * FROM
460    TABLE(QSYS2.PARSE_STATEMENT('SELECT nom, srv, nomsrv
461          FROM employes INNER JOIN services USING(srv)')) AS T;
462
```

NAME_TYPE	NAME	SCHEMA	RDB	COLUMN_NAME	USAGE_TYPE	NAME_START_POSITION	SQL_STATEMENT_TYPE
COLUMN	-	-	-	NOM	QUERY	8	QUERY
COLUMN	-	-	-	SRV	QUERY	13	QUERY
COLUMN	-	-	-	NOMSRV	QUERY	18	QUERY
TABLE	EMPLOYES	-	-	-	QUERY	30	QUERY
TABLE	SERVICES	-	-	-	QUERY	50	QUERY

Exemple de service sous forme d'UDTF

- Combinaison des services SYSPROGRAMSTMTSTAT et PARSE_STATEMENT

```
463 SELECT PROGRAM_NAME, T.*
464    FROM QSYS2.SYSPROGRAMSTMTSTAT, TABLE(QSYS2.PARSE_STATEMENT(STATEMENT_TEXT)) AS T
465    WHERE PROGRAM_SCHEMA = 'AS425F';
466
467
```

Les "objets" SQL manipulés par les programmes de la bibliothèque AS425F

PROGRAM_NAME	NAME_TYPE	NAME	SCHEMA	RDB	COLUMN_NAME	USAGE_TYPE	NAME_START_POSITION	SQL_STATEMENT_TYPE
AUTORENS	TABLE	TABEMPL3	AS425F	-	-	TARGET TA...	13	INSERT
AUTORENS	COLUMN	TABEMPL3	AS425F	-	NOM	TARGET TA...	33	INSERT
AUTORENS	COLUMN	TABEMPL3	AS425F	-	SEXE	TARGET TA...	39	INSERT
AUTORENS	COLUMN	TABEMPL3	AS425F	-	NUMSRV	TARGET TA...	46	INSERT
AUTORENS	COLUMN	TABEMPL3	AS425F	-	SALAIRE	TARGET TA...	55	INSERT
AUTORENS	COLUMN	TABEMPL3	AS425F	-	DATENAI	TARGET TA...	65	INSERT
BLOBRPG1	TABLE	TABEMP2	-	-	-	TARGET TA...	8	UPDATE
BLOBRPG1	COLUMN	TABEMP2	-	-	CV	TARGET TA...	20	UPDATE
BLOBRPG1	COLUMN	-	-	-	MAT	QUERY	35	UPDATE

Les UDTFs fournies dans la bibliothèque SYSTOOLS

- Fonctions HTTP
- Permettent d'accéder à des ressources Web
  - Pages Web, flux XML, flux RSS, photos, PDF, Services Web…
  - Sont disponibles sous forme d'UDFs et UDTFs
- Les UDTFs
  - HTTP*VERBOSE
  - Renvoient une table contenant une ligne, avec une colonne pour la valeur de l'entête HTTP et une colonne pour la valeur résultat

```
7  SELECT * FROM TABLE(SYSTOOLS.HTTPGETCLOBVERBOSE('https://www.ibm.com/fr', '')) AS res;
```

RESPONSEMSG	RESPONSEHTTPHEADER
<!DOCTYPE html><html lang="fr-FR"><h...	<?xml version="1.0" encoding="UTF-8" ?><httpHeader responseCode="200"><...


MERCI

Annexe

Les UDTFs externes

Les UDTFs externes

- Sont créées par l'instruction SQL CREATE FUNCTION pour définir :
  - Les paramètres en entrée
  - Les colonnes en retour
  - Les options
  - Le nom du programme ou de la procédure de programme de service invoqué

- Le programme ou la procédure
  - Reçoit des paramètres en entrée : les paramètres de la procédure + des attributs indiquant ce que doit faire le programme (ouverture, lecture d'une ligne, fermeture…)
  - Renvoie une ligne et/ou un code retour (succès, fin de fichier, erreur)

Les UDTFs externes

Options Unique to External Table Functions

NO DBINFO or DBINFO

- Specifies whether additional information structure is passed to the table function.

PROGRAM TYPE MAIN or PROGRAM TYPE SUB

- Specifies whether the “external program” is a program or service program.

NO FINAL CALL or FINAL CALL

- Specifies whether a separate FIRST and FINAL call is made to the “external program”.

NO SCRATCHPAD or SCRATCHPAD

- Specifies whether a scratchpad is provided and passed to the “external program”.

EXTERNAL or EXTERNAL NAME

- Specifies the name of the “external program” (program name or service program entry point).

Les UDTFs externes

Implementing an External Table Function – SQL Statements

```
CREATE OR REPLACE FUNCTION DEPTEMPLOYEES3 (DEPTNO CHAR(3))
        RETURNS TABLE (EMPNO CHAR(6), LASTNAME VARCHAR(15), FIRSTNME
VARCHAR(12))
DETERMINISTIC
LANGUAGE C
SPECIFIC DEPTEMP3
NOT DETERMINISTIC
READS SQL DATA
CALLED ON NULL INPUT
ALLOW PARALLEL
FINAL CALL
NOT FENCED
SCRATCHPAD 1000
CARDINALITY 32767
EXTERNAL NAME 'MJATST/MYSRVPGM(DEPTEMPLOYEES3)'
PARAMETER STYLE DB2SQL ;
```

More Information:

- SQL Reference - **CREATE FUNCTION (External Table)**
- SQL Programming - Writing UDFs as external functions

Les UDTFs externes

Parameters Passed to the External Table Function Program

- The first N parameters are the input parameters specified on the CREATE FUNCTION statement.
- The next M parameters are the result columns of the function specified on the RETURNS TABLE clause.
- N parameters for indicator variables for the input parameters.
- M parameters for the indicator variables of the result columns of the function
- A CHAR(5) output parameter for SQLSTATE. The SQLSTATE returned indicates the success or failure of the function.
- A VARCHAR(517) input parameter containing the fully qualified function name.
- A VARCHAR(128) input parameter containing the specific name.
- A VARCHAR(1000) output parameter for the message text.
- A structure (consisting of an INTEGER followed by a CHAR(n)) input and output parameter for the scratchpad, if SCRATCHPAD was specified.
- An INTEGER input parameter for the call type.
- The dbinfo structure, if DBINFO was specified.

These parameters are passed according to the specified LANGUAGE. For example, if the language is C or C++, the VARCHAR parameters are passed as NUL-terminated strings. For more information about the parameters passed, see the include sqludf in the appropriate source file in library QSYSINC. For example, for C, sqludf can be found in QSYSINC/H.

More Information

- SQL Programming - Parameter passing conventions for stored procedures and user-defined functions

Les UDTFs externes

Parameters Passed Example

- INPUT: The first parameter will contain the DEPTNO input value.
- OUTPUT: 3 parameters are the returned values for the EMPNO, LASTNAME, and FIRSTNME result columns.
- INPUT: 1 parameters for the input DEPTNO indicator variable.
- OUTPUT: 3 parameters for the EMPNO, LASTNAME, and FIRSTNME indicator variables of the result columns.
- OUTPUT: 1 CHAR(5) parameter for SQLSTATE.
- INPUT: 1 VARCHAR(517) input parameter containing the fully qualified function name.
- INPUT: 1 VARCHAR(128) parameter containing the specific name.
- OUTPUT: 1 VARCHAR(1000) output parameter for the message text.
- INPUT/OUTPUT: 1 parameter for the scratchpad (since we did specify a SCRATCHPAD).
- INPUT: 1 INTEGER input parameter for the call type.

Note that DBINFO was not specified on the CREATE FUNCTION statement so no input parameter was passed with the DBINFO structure.

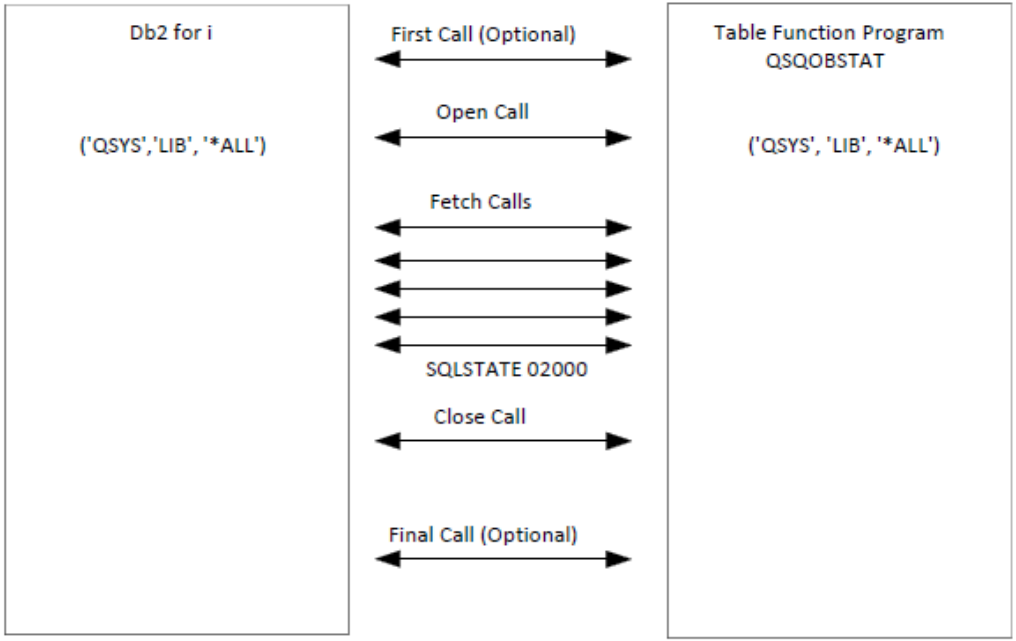
More Information

- SQL Programming - Parameter passing conventions for stored procedures and user-defined functions

Les UDTFs externes

How Does an External Table Function Work

```
SELECT *
FROM TABLE
(qsys2.object_statistics(
'QSYS', 'LIB', '*ALL')) AS a;
```



Les UDTFs externes

Implementing a View That Uses Lateral Correlation

```
CREATE VIEW SYSPARTITIONSTAT (
        Table_Schema                    FOR COLUMN Tabschema,
        Table_Name                      FOR COLUMN Tabname,
        Table_Partition                 FOR COLUMN Tabpart,
        Partition_Type                  FOR COLUMN PARTTYPE,
        Partition_Number                FOR COLUMN PARTNBR,
        Number_Distributed_Partitions   FOR COLUMN DSTPARTS,
        Number_Rows                     FOR COLUMN CARD,  ...   )
        AS
        SELECT  A.DBXLB2,
                A.DBXLFI,
                COALESCE(B.Table_Partition,''),
                CHAR(COALESCE(B.Partition_type,' '),1),
                B.Partition_number,
                B.Number_distributed_partitions,
                COALESCE(B.Number_Rows,0),
                ...
                COALESCE(B.Media_Preference,SMALLINT(0)),
                A.DBXLIB AS System_table_schema,
                A.DBXFIL AS System_table_name,
                CHAR(COALESCE(B.System_Table_Member,''),10)
        FROM QSYS.QADBXREF AS A,
        LATERAL ( SELECT * FROM TABLE (Partition_Statistics(A.DBXLIB, A.DBXFIL) ) AS X ) AS B
        WHERE a.DBXATR IN ('TB','PF','MQ') AND a.DBXREL = 'Y' AND B.Partition_number <> 0
```

Les UDTFs externes

How Does an External Table Function Work with Lateral Correlation

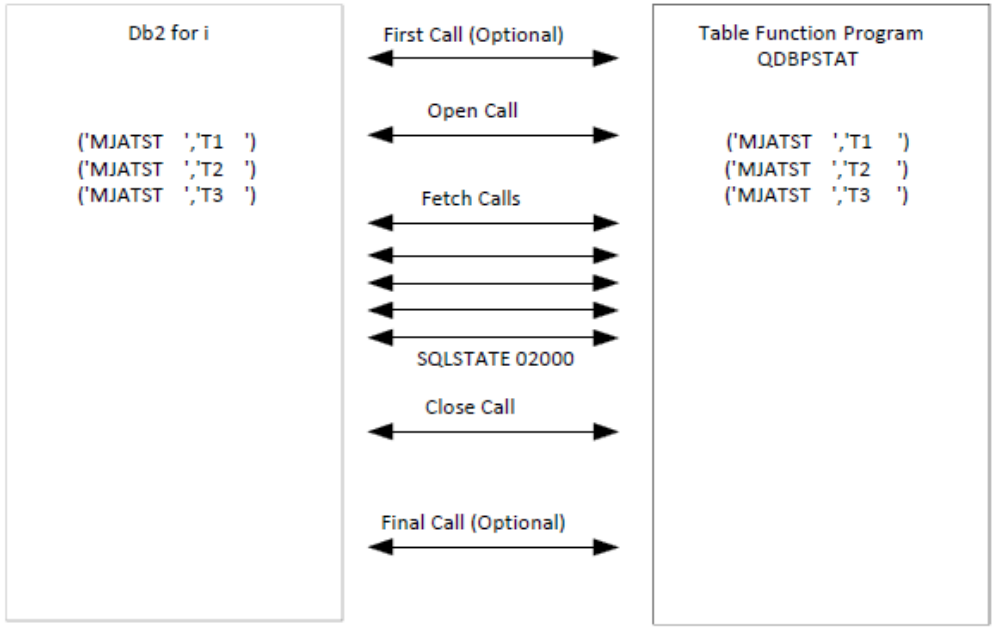
```
SELECT *
FROM qsys2.syspartitionstat
WHERE table_schema = 'MJATST'
```

Inside SYSPARTITIONSTAT:

```
SELECT ...

FROM qsys.qadbxref AS a,
LATERAL (SELECT * FROM TABLE
(Partition_Statistics(A.DBXLIB, A.DBXFIL) )
AS x ) AS b

WHERE a.dbxatr IN ('TB','PF','MQ')
AND a.dbxrel = 'Y'
AND b.partition_number <> 0
```



Les UDTFs externes

Best Practices

- Use ILE languages
- Consider performance when determining what parameters to define in your table functions
- Use variable length instead of fixed length for parameters
- Give the function a longer meaningful name and a short 10 byte specific name
- Use parameter DEFAULTS , if applicable
- Grant privileges based on your security requirements
- Return nulls for values that are not applicable
- Provide views with lateral correlation both a meaningful short and long column name
- Consider performance when running queries on views with lateral correlation.

UDTFs externes – Pour en savoir plus