121

DB2

November 2002

In this issue

update

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

DB2 Update on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

User-defined functions

Here is a PL/I program that I used to get acquainted with DB2 user-defined functions.

This program can behave as a scalar function and a table function according to the specific name under which it is called.

HVPUDF1 PL/I SOURCE

```
/*HVPUDF1 DB2 USER DEFINED FUNCTION UDF*/
 /* LANGUAGE(PLI) SYSTEM(MVS) DB2(YES) DB2AFI(DSNRLI)*/
 /* LKDNCLD(HVPC2X)*/
*PROCESS SYSTEM(MVS);
 HVPUDF1:PROC(UDF_PARM1,UDF_RESULT,
              UDF_IND1,UDF_INDR,
              UDF_SQLSTATE,UDF_NAME,UDF_SPEC_NAME,
              UDF_DIAG_MSG,UDF_SCRATCHPAD,
              UDF_CALL_TYPE,DBINFO)
             OPTIONS(MAIN NOEXECOPS REENTRANT);
 /* DOCUM BEGIN
  OTHER EXAMPLES
   DSN61Ø.SDSNSAMP(DSN8DUWF)
  HOW TO CALL:
   SELECT HVPUDF1('MV      99992ØØ12ØØ2Ø96962ØØ2-Ø7-Ø1B????')
    AS FIELD1 FROM SYSIBM.SYSDUMMY1
   SELECT FIELD1 FROM
    TABLE(HVPUDF2('MV      99992ØØ12ØØ2Ø96962ØØ2-Ø7-Ø1B????'))
    AS QQ

 WLM,DB2
   V WLM,APPLENV=DBPRCRX1,RESUME
   -DISPLAY FUNCTION SPECIFIC(TSHVR.HVPUDF1)
   -START FUNCTION SPECIFIC(TSHVR.HVPUDF1)
   -START FUNCTION SPECIFIC(TSHVR.HVPUDF2)
   -STOP  FUNCTION SPECIFIC(TSHVR.HVPUDF1)
   -STOP  FUNCTION SPECIFIC(TSHVR.HVPUDF2)
   V WLM,APPLENV=DBPRCRX1,Q

 DB2 USES THE RECOVERABLE RESOURCE MANAGER SERVICES ATTACHMENT
  FACILITY (RRSAF) AS ITS INTERFACE WITH USER-DEFINED FUNCTION,
  SO USE DSNRLI AS DB2 LANGUAGE INTERFACE

  ERRORS
  DSNT4Ø8I SQLCODE=-471 INVOCATION OF FUNCTION OR PROCEDURE
           TSHVR.HVPUDF1 FAILED DUE TO REASON ØØE79ØØ1
```

```
   -START FUNCTION COMMAND TO ACTIVATE THE USER-DEFINED FUNCTION

 DSNT4Ø8I SQLCODE=-44Ø  NO FUNCTION BY THE NAME HVPUDF1 HAVING
  COMPATIBLE ARGUMENTS WAS FOUND IN THE CURRENT PATH
 DSNT4Ø8I SQLCODE=-444 USER PROGRAM HVPUDF1 COULD NOT BE FOUND
 DSNT4Ø8I SQLCODE=-43Ø FUNCTION TSHVR.HVPUDF1
 (SPECIFICN AME TSHVR.HVPUDF1) HAS ABNORMALLY TERMINATED

 DSNX9Ø6I - DSNX9CAC PROCEDURE OR FUNCTION TSHVR.HVPUDF1
 TERMINATED ABNORMALLY. THE PROCEDURE OR FUNCTION HAS BEEN STOPPED.
 ASID= Ø04D WLM_ENV= DBPRCRX1 DSN32Ø1I - ABNORMAL EOT IN PROGRESS FOR
 USER=DEFAULT  CONNECTION-ID=RRSAF CORRELATION-ID=
 JOBNAME=DBPRCRX1 TCB=ØØBE918Ø

   DOCUM END
*/
%INCLUDE BUILTIN;
EXEC SQL INCLUDE SQLCA;
/*EXEC SQL BEGIN DECLARE SECTION*/
DCL 1 APMVRØ1,
       5 LEGMVJ1          BIN FIXED(15),
       5 LEGMVJ2          BIN FIXED(15),
       5 ZITNGMV          CHAR(1),
       5 INTERMV          CHAR(4),
       5 M9TXTN11         CHAR(75),
       5 M9TXTN12         CHAR(75),
       5 M9TXTN13         CHAR(75),
       5 M9TXTN14         CHAR(75),
       5 M9TXTN15         CHAR(75),
       5 M9TXTN16         CHAR(75),
       5 M9TXTN17         CHAR(75),
       5 M9TXTN18         CHAR(75),
       5 M9TXTN19         CHAR(75),
       5 M9TXTF11         CHAR(75),
       5 M9TXTF12         CHAR(75),
       5 M9TXTF13         CHAR(75),
       5 M9TXTF14         CHAR(75),
       5 M9TXTF15         CHAR(75),
       5 M9TXTF16         CHAR(75),
       5 M9TXTF17         CHAR(75),
       5 M9TXTF18         CHAR(75),
       5 M9TXTF19         CHAR(75);
 DCL LEGISJ1 BIN FIXED(15);
 DCL LEGISJ2 BIN FIXED(15);
 DCL ZITTING CHAR(1);
 DCL NR      CHAR(4);
/*EXEC SQL END DECLARE SECTION*/

EXEC SQL DECLARE XXXX.YY TABLE
          ( LEGMVJ1                       SMALLINT NOT NULL,
            LEGMVJ2                       SMALLINT NOT NULL,
```

```
 ZITNGMV                        CHAR(1) NOT NULL,
 INTERMV                        CHAR(4) NOT NULL,
 VERTYPE                        CHAR(1) NOT NULL,
 CODECOM                        CHAR(4) NOT NULL,
 NHANDEL                        CHAR(4) NOT NULL,
"NBLADZ$"                       CHAR(5) NOT NULL,
 NVRAGER                        CHAR(5) NOT NULL,
 DVERGAD                        DATE,
 MVPARTY                        CHAR(6) NOT NULL,
"AARDVR$"                       CHAR(1) NOT NULL,
"NMINVR$1"                      CHAR(5) NOT NULL,
"NMINVR$2"                      CHAR(5) NOT NULL,
 MVLEGIV1                       CHAR(2) NOT NULL,
 MVLEGIV2                       CHAR(2) NOT NULL,
 MVDEPTV1                       SMALLINT NOT NULL,
 MVDEPTV2                       SMALLINT NOT NULL,
 MVPARTV1                       CHAR(6) NOT NULL,
 MVPARTV2                       CHAR(6) NOT NULL,
"NMINAN$1"                      CHAR(5) NOT NULL,
"NMINAN$2"                      CHAR(5) NOT NULL,
 MVLEGIA1                       CHAR(2) NOT NULL,
 MVLEGIA2                       CHAR(2) NOT NULL,
 MVDEPTA1                       SMALLINT NOT NULL,
 MVDEPTA2                       SMALLINT NOT NULL,
 MVPARTA1                       CHAR(6) NOT NULL,
 MVPARTA2                       CHAR(6) NOT NULL,
"OPMVRA$"                       CHAR(75) NOT NULL,
 M9TXTN11                       CHAR(75) NOT NULL,
 M9TXTN12                       CHAR(75) NOT NULL,
 M9TXTN13                       CHAR(75) NOT NULL,
 M9TXTN14                       CHAR(75) NOT NULL,
 M9TXTN15                       CHAR(75) NOT NULL,
 M9TXTN16                       CHAR(75) NOT NULL,
 M9TXTN17                       CHAR(75) NOT NULL,
 M9TXTN18                       CHAR(75) NOT NULL,
 M9TXTN19                       CHAR(75) NOT NULL,
 M9TXTF11                       CHAR(75) NOT NULL,
 M9TXTF12                       CHAR(75) NOT NULL,
 M9TXTF13                       CHAR(75) NOT NULL,
 M9TXTF14                       CHAR(75) NOT NULL,
 M9TXTF15                       CHAR(75) NOT NULL,
 M9TXTF16                       CHAR(75) NOT NULL,
 M9TXTF17                       CHAR(75) NOT NULL,
 M9TXTF18                       CHAR(75) NOT NULL,
 M9TXTF19                       CHAR(75) NOT NULL,
 BDELETE                        CHAR(1) NOT NULL,
 CONCUPD                        SMALLINT NOT NULL,
 KUSERID                        CHAR(8) NOT NULL,
 DUPDATE                        DATE,
 DCREATE                        DATE
) ;
```

```
EXEC SQL DECLARE CURSOR1 CURSOR FOR
 SELECT
              LEGMVJ1,
              LEGMVJ2,
              ZITNGMV,
              INTERMV,
              M9TXTN11,
              M9TXTN12 ,
              M9TXTN13 ,
              M9TXTN14 ,
              M9TXTN15 ,
              M9TXTN16 ,
              M9TXTN17 ,
              M9TXTN18 ,
              M9TXTN19 ,
              M9TXTF11 ,
              M9TXTF12 ,
              M9TXTF13 ,
              M9TXTF14 ,
              M9TXTF15 ,
              M9TXTF16 ,
              M9TXTF17 ,
              M9TXTF18 ,
              M9TXTF19
       FROM XXXX.YY
       WHERE LEGMVJ1 >= :LEGISJ1
         AND LEGMVJ2 >= :LEGISJ2
         AND ZITNGMV >= :ZITTING
         AND INTERMV >= :NR;
         /*AND DUPDATE > :UPDATE_DATE*/

DCL $SQL_ROW_NOT_FOUND BIN FIXED(31) STATIC INIT(100);
/*SEE ALSO DSN610.ADSNMACS(SQLUDF)*/
DCL $SQLUDF_TF_FIRST BIN FIXED STATIC INIT(-2);
DCL $SQLUDF_TF_OPEN  BIN FIXED STATIC INIT(-1);
DCL $SQLUDF_TF_FETCH BIN FIXED STATIC INIT(0);
DCL $SQLUDF_TF_CLOSE BIN FIXED STATIC INIT(1);
DCL $SQLUDF_TF_FINAL BIN FIXED STATIC INIT(2);
DCL $SQLUDF_TF_FINAL_CRA BIN FIXED STATIC INIT(255);

DCL $CALL_FIRST_SCALAR  BIN FIXED STATIC INIT(-1);
DCL $CALL_NORMAL_SCALAR BIN FIXED STATIC INIT(0);
DCL $CALL_FINAL_SCALAR  BIN FIXED STATIC INIT(1);
DCL $CALL_FINAL_COMMIT_SCALAR  BIN FIXED STATIC INIT(255);

DCL UDF_PARM1 CHAR(*) VARYING; /* FIRST PARAMETER */
DCL UDF_RESULT CHAR(*) VARYING ; /* RESULT PARAMETER */
DCL UDF_IND1 BIN FIXED(15); /* INDICATOR FOR 1ST PARM */
DCL UDF_INDR BIN FIXED(15); /* INDICATOR FOR RESULT */
DCL UDF_SQLSTATE CHAR(5); /* SQLSTATE RETURNED TO DB2 */
DCL UDF_NAME CHAR(137) VARYING; /* QUALIFIED FUNCTION NAME */
```

```
DCL UDF_SPEC_NAME CHAR(128) VARYING; /* SPECIFIC FUNCTION NAME */
DCL UDF_DIAG_MSG CHAR(70) VARYING; /* DIAGNOSTIC STRING */
/*UDF_SCRATCHPAD:
  IF FUNCTION DEFINED WITH NO SCRATCHPAD THEN PARAMETER IS NOT
  PASSED ||
  TO TEST FOR VALID SCRATCHPAD ONE SHOULD BE ABLE TO TEST
  NUMBER OF PARAMETERS PASSED TO PLI
  IF NO SCRATCHPAD PASSED THEN UDF_CALL_TYPE TAKES VALUE OF
  NEXT PARM WHICH IS DBINFO-POINTER
*/
DCL 01 UDF_SCRATCHPAD,
 03 UDF_SPAD_LEN BIN FIXED(31),
 03 UDF_SPAD_TEXT CHAR(100);
DCL UDF_CALL_TYPE BIN FIXED(31); /* CALL TYPE */
DCL DBINFO POINTER;
/* CONSTANTS FOR DB2_ENCODING_SCHEME */
DCL SQLUDF_ASCII BIN FIXED(15) INIT(1);
DCL SQLUDF_EBCDIC BIN FIXED(15) INIT(2);
DCL SQLUDF_MIXED BIN FIXED(15) INIT(3);
DCL 01 UDF_DBINFO BASED(DBINFO) UNALIGNED,
 03 UDF_DBINFO_LLEN BIN FIXED(15), /* LOCATION LENGTH */
 03 UDF_DBINFO_LOC CHAR(128), /* LOCATION NAME */
 03 UDF_DBINFO_ALEN BIN FIXED(15), /* AUTH ID LENGTH */
 03 UDF_DBINFO_AUTH CHAR(128), /* AUTHORIZATION ID */
 03 UDF_DBINFO_CDPG,
  05 DB2_CCSIDS(3),
   07 R1 BIN FIXED(15),
   07 DB2_SBCS BIN FIXED(15),
   07 R2 BIN FIXED(15),
   07 DB2_DBCS BIN FIXED(15),
   07 R3 BIN FIXED(15),
   07 DB2_MIXED BIN FIXED(15),
  05 DB2_ENCODING_SCHEME BIN FIXED(31),
  05 DB2_CCSID_RESERVED CHAR(8),
 03 UDF_DBINFO_SLEN BIN FIXED(15),
 03 UDF_DBINFO_SCHEMA CHAR(128),
 03 UDF_DBINFO_TLEN BIN FIXED(15),
 03 UDF_DBINFO_TABLE CHAR(128),
 03 UDF_DBINFO_CLEN BIN FIXED(15),
 03 UDF_DBINFO_COLUMN CHAR(128),
 03 UDF_DBINFO_RELVER CHAR(8),
 03 UDF_DBINFO_PLATFORM BIN FIXED(31),
 03 UDF_DBINFO_NUMTFCOL BIN FIXED(15),
 03 UDF_DBINFO_RESERV1 CHAR(24),
 03 UDF_DBINFO_TFCOLUMN PTR,
 03 UDF_DBINFO_APPLID PTR,
 03 UDF_DBINFO_RESERV2 CHAR(20);

 DCL $FALSE BIT(1) STATIC INIT('0'B);
 DCL $TRUE  BIT(1) STATIC INIT('1'B);
```

```
DCL MYRC BIN FIXED(31) INIT(Ø);
DCL TRCLVL BIN FIXED(31) INIT(Ø);
DCL MYRC_PIC PIC'99999';
DCL CURRENT_PATH CHAR(254) VARYING;
DCL CURRENT_PACKAGESET CHAR(18);
DCL SYSPRINT FILE STREAM OUTPUT;
DCL HVPC2X ENTRY OPTIONS(INTER,ASM,RETCODE);
DCL PARM1 CHAR(4Ø);
DCL Ø1 PARM1_REDEF BASED(ADDR(PARM1)),
    Ø2 PTBL   CHAR(8),/*MV*/
    Ø2 PTRCLVL PIC '9999',
    Ø2 PJ1    PIC '9999',
    Ø2 PJ2    PIC '9999',
    Ø2 PZ     CHAR(1),
    Ø2 PNR    CHAR(4),
    Ø2 PUPDATE CHAR(1Ø),
    Ø2 PLANG  CHAR(1),
    Ø2 POTH   CHAR(4);
DCL SQLCODE_PIC PIC '99999';

EXEC SQL SET :CURRENT_PATH=CURRENT PATH;
EXEC SQL SET :CURRENT_PACKAGESET=CURRENT PACKAGESET;

PUT SKIP EDIT('BEGIN DATE=',DATETIME(),
              ' UDF_CALL_TYPE=',UDF_CALL_TYPE)(A,A,A,F(5));
PUT SKIP EDIT('UDF_SQLSTATE=',UDF_SQLSTATE)(A);

MYRC=CALL_TYPE_CHECK();
IF MYRC¬=Ø THEN DO;
 GOTO L_MAIN_EXIT;
END;

UDF_SQLSTATE='ØØØØØ';
UDF_INDR=-1;

IF UDF_SPEC_NAME='HVPUDF2' THEN DO;
 CALL HVPUDF1_TF;
END;/*TABLE FUNCTION*/
ELSE IF UDF_SPEC_NAME='HVPUDF1' THEN DO;
 MYRC=HVPUDF1_SCALAR();
END;/*SCALAR FUNCTION*/
ELSE DO;
 MYRC=386Ø2;
 UDF_SQLSTATE='386Ø2';
 UDF_DIAG_MSG='UNEXPECTED UDF_SPEC_NAME '||UDF_SPEC_NAME;
END;
L_MAIN_EXIT:
IF MYRC¬=Ø THEN DO;
 MYRC_PIC=MYRC;
 IF UDF_SQLSTATE='ØØØØØ' THEN  UDF_SQLSTATE='386Ø2';
 IF UDF_DIAG_MSG='' THEN
```

```
    UDF_DIAG_MSG='UNEXPECTED MYRC '||MYRC_PIC;
 END;
 PUT SKIP EDIT('UDF_SQLSTATE=',UDF_SQLSTATE)(A);
 PUT SKIP EDIT('END DATE=',DATETIME(),
              ' UDF_CALL_TYPE=',UDF_CALL_TYPE)(A,A,A,F(5));
 PUT SKIP LIST('');
 RETURN(MYRC);
 /**/
S_DEBUG_FIRST:PROCEDURE;
 DCL PTR_HEX CHAR(8);
 DCL SPAD_HEX CHAR(16);
 IF TRCLVL=Ø THEN RETURN;
 PUT SKIP EDIT('CURRENT PATH=',CURRENT_PATH)(A);
 /*CURRENT PATH="SYSIBM","SYSFUN","SYSPROC","TSHVR"*/
 PUT SKIP EDIT('CURRENT PACKAGESET=',CURRENT_PACKAGESET)(A);
 PUT SKIP EDIT('UDF_NAME=',UDF_NAME)(A);
 PUT SKIP EDIT('UDF_SPEC_NAME=',UDF_SPEC_NAME)(A);
 /*UDF_SPEC_NAME=HVPUDF1*/
 PUT SKIP EDIT('UDF_IND1=',UDF_IND1)(A,F(4));
 PUT SKIP EDIT('UDF_INDR=',UDF_INDR)(A,F(4));
 PUT SKIP EDIT('UDF_PARM1 LENGTH=',LENGTH(UDF_PARM1))(A,F(4));
 IF (LENGTH(UDF_PARM1)>Ø) & (UDF_IND1>=Ø)  THEN DO;
  PUT SKIP EDIT('UDF_PARM1=',UDF_PARM1)(A);
 END;
 PUT SKIP EDIT('UDF_RESULT LENGTHS=',
              STG(UDF_RESULT),CSTG(UDF_RESULT),LENGTH(UDF_RESULT))
              (A,3(F(4)));
 PUT SKIP EDIT('UDF_DBINFO_ALEN=',UDF_DBINFO_ALEN)(A,F(4));
 PUT SKIP EDIT('UDF_DBINFO_LLEN=',UDF_DBINFO_LLEN)(A,F(4));
 PUT SKIP EDIT('UDF_DBINFO_SLEN=',UDF_DBINFO_SLEN)(A,F(4));
 PUT SKIP EDIT('UDF_DBINFO_TLEN=',UDF_DBINFO_TLEN)(A,F(4));
 CALL HVPC2X(UDF_DBINFO_TFCOLUMN,PTR_HEX,BINARY(4,31));
 PUT SKIP EDIT('UDF_DBINFO_TFCOLUMN=',PTR_HEX)(A);
 CALL HVPC2X(ADDR(UDF_SCRATCHPAD),PTR_HEX,BINARY(4,31));
 PUT SKIP EDIT('UDF_SCRATCHPAD PTR=',PTR_HEX)(A);
 PUT SKIP EDIT('UDF_SPAD_LEN=',UDF_SPAD_LEN)(A,F(4));
END S_DEBUG_FIRST;
/**/
CALL_TYPE_CHECK:PROCEDURE RETURNS(BIN FIXED(31));
 DCL MYRC BIN FIXED(31) INIT(Ø);
 DCL VERY_FIRST BIT(1) INIT($FALSE);
 IF UDF_SPEC_NAME='HVPUDF1' THEN DO;
  IF UDF_DBINFO_TFCOLUMN¬=SYSNULL THEN DO;
   PUT SKIP EDIT('SCALAR AND TFCOLUMN<>SYSNULL')(A);
  END;
  IF UDF_CALL_TYPE=$CALL_FIRST_SCALAR THEN VERY_FIRST=$TRUE;
  ELSE IF UDF_CALL_TYPE=$CALL_NORMAL_SCALAR THEN;
  ELSE IF UDF_CALL_TYPE=$CALL_FINAL_SCALAR THEN;
  ELSE IF UDF_CALL_TYPE=$CALL_FINAL_COMMIT_SCALAR THEN;
  ELSE DO;
   MYRC=16;
```

```
  PUT SKIP EDIT('SCALAR UDF_CALL_TYPE=',UDF_CALL_TYPE)(A,F(5));
 END;
END;/*HVPUDF1*/
ELSE IF UDF_SPEC_NAME='HVPUDF2' THEN DO;
 /*
 UDF_DBINFO_TFCOLUMN ALWAYS SEEMS TO BE SYSNULL|
 IF UDF_DBINFO_TFCOLUMN=SYSNULL THEN DO
  PUT SKIP EDIT('TABLE AND TFCOLUMN=SYSNULL')(A)
 END
 */
 IF UDF_CALL_TYPE=$SQLUDF_TF_FIRST THEN VERY_FIRST=$TRUE;
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_OPEN THEN;
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_FETCH THEN;
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_CLOSE THEN;
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_FINAL THEN;
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_FINAL_CRA THEN;
 ELSE DO;
  MYRC=16;
  PUT SKIP EDIT('TABLE  UDF_CALL_TYPE=',UDF_CALL_TYPE)(A,F(5));
 END;
END;/*HVPUDF2*/
ELSE DO;
 UDF_DIAG_MSG='UNKNOWN UDF_SPEC_NAME';
 MYRC=16;
END;
IF MYRC¬=Ø THEN GOTO L_CALL_TYPE_CHECK_EXIT;

/*UDF_PARM1 SEEMS TO BE FILLED ON ALL CALLS */
IF (LENGTH(UDF_PARM1)=Ø) | (UDF_IND1<Ø)  THEN DO;
 PUT SKIP EDIT('LENGTH(UDF_PARM1)=',LENGTH(UDF_PARM1),
               ' UDF_IND1=',UDF_IND1)(A,F(5),A,F(5));
END;
IF LENGTH(UDF_PARM1)¬=STG(PARM1) THEN DO;
 UDF_DIAG_MSG='INVALID UDF_PARM1 '||UDF_PARM1;
 MYRC=16;
 GOTO L_CALL_TYPE_CHECK_EXIT;
END;
IF UDF_SPAD_LEN<1ØØ THEN DO;
 MYRC=386Ø2;
 UDF_SQLSTATE='386Ø2';
 UDF_DIAG_MSG='INVALID UDF_SPAD_LEN';
 GOTO L_CALL_TYPE_CHECK_EXIT;
END;
PARM1=UDF_PARM1;
/*STILL TO DO:CHECK VALIDITY OF INPUT FIRST|| NUMERIC ETC..*/
TRCLVL=PTRCLVL;
IF VERY_FIRST=$TRUE THEN DO;
 CALL S_DEBUG_FIRST;
END;
L_CALL_TYPE_CHECK_EXIT:
RETURN(MYRC);
```

```
END CALL_TYPE_CHECK;
/**/
SQLCODE_CHECK:PROCEDURE(MSG) RETURNS(BIN FIXED(31));
 DCL MSG CHAR(*);
 DCL MYRC BIN FIXED(31);

 IF SQLCODE=Ø THEN RETURN(SQLCODE);
 PUT SKIP EDIT(MSG,' SQLCODE=',SQLCODE)(A,A,F(5));
 SQLCODE_PIC=SQLCODE;
 RETURN(SQLCODE);
END SQLCODE_CHECK;
/**/
RTRIM:PROC(NAME,LASTNAME) RETURNS(CHAR(255) VARYING);
/* FROM TSHVR.SOURCE.PROD(RTRIM)*/
 DCL NAME CHAR(*);
 DCL LASTNAME BIT(1);
 DCL NAMEL BIN FIXED(15);
 DCL LASTCHAR CHAR(1) INIT('');
 DCL RTRIMNAME CHAR(255) VARYING   ;
 DCL (LOPER) BIN FIXED;

 NAMEL=LENGTH(NAME);
 /***
 IF NAMEL>Ø THEN DO
  LASTCHAR=SUBSTR(NAME,NAMEL,1)
 END
 ***/
 DO LOPER=NAMEL TO 1 BY -1;
  IF SUBSTR(NAME,LOPER,1) ¬=' ' THEN  LEAVE;
 END;/*DO LOPER=NAMEL TO 1*/
 IF LOPER>Ø THEN DO;
  LASTCHAR=SUBSTR(NAME,LOPER,1);
  IF LASTCHAR='-' THEN DO;
   IF LOPER>1 THEN LOPER=LOPER-1;
  END;
  ELSE DO;
   IF LASTNAME=$FALSE THEN DO;
    IF LOPER<NAMEL THEN LOPER=LOPER+1;/*+EXTRA BLANKO*/
   END;
  END;
  RTRIMNAME=SUBSTR(NAME,1,LOPER);
 END;
 ELSE DO;
  RTRIMNAME='';
 END;
 RETURN(RTRIMNAME);
END RTRIM;
/**/
HVPUDF1_TF:PROCEDURE;
 /* CALLED AS TABLE FUNCTION */
 DCL MYRC BIN FIXED(31) INIT(Ø);
```

```
 DCL TELLER BIN FIXED(31) BASED(ADDR(UDF_SPAD_TEXT));
 IF TELLER>1ØØ THEN DO;
  MYRC=386Ø2;
  UDF_SQLSTATE='386Ø2';
  DCL TELLER_PIC PIC '99999';
  TELLER_PIC=TELLER;
  UDF_DIAG_MSG='UNEXPECTED TELLER '||TELLER_PIC;
  GOTO HVPUDF1_TF_EXIT;
 END;
 TELLER=TELLER+1;
 IF UDF_CALL_TYPE=$SQLUDF_TF_OPEN THEN DO;
  LEGISJ1=PJ1;
  LEGISJ2=PJ2;
  ZITTING=PZ;
  NR     =PNR;
  EXEC SQL OPEN CURSOR1;
  MYRC=SQLCODE_CHECK('OPEN CURSOR1');
 END;/*$SQLUDF_TF_OPEN*/
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_FETCH THEN DO;
  EXEC SQL FETCH CURSOR1 INTO :APMVRØ1;
  IF SQLCODE=$SQL_ROW_NOT_FOUND THEN DO;
   UDF_SQLSTATE='Ø2ØØØ';
  END;/*EOF*/
  ELSE DO;
   MYRC=SQLCODE_CHECK('FETCH CURSOR1');
   IF MYRC=Ø THEN DO;
    DCL Ø1 IDENT,
     Ø2 OJ1    PIC '9999',
     Ø2 OJ2    PIC '9999',
     Ø2 OZ     CHAR(1),
     Ø2 ONR    CHAR(4);
    /*OUTPUT*/
    OJ1=LEGMVJ1;
    OJ2=LEGMVJ2;
    OZ=ZITNGMV;
    ONR=INTERMV;
    UDF_INDR=Ø;
    UDF_RESULT=STRING(IDENT)||TEXTN()||'@@@'||TEXTF();
   END;
   ELSE DO;
    UDF_SQLSTATE='386Ø2';
    UDF_DIAG_MSG='UNEXPECTED SQLCODE '||SQLCODE_PIC;
   END;
  END;/*NOT EOF*/
 END;/*$SQLUDF_TF_FETCH*/
 ELSE IF UDF_CALL_TYPE=$SQLUDF_TF_CLOSE THEN DO;
  EXEC SQL CLOSE CURSOR1;
  MYRC=SQLCODE_CHECK('CLOSE CURSOR1');
 END;/*$SQLUDF_TF_CLOSE*/
 HVPUDF1_TF_EXIT:
END HVPUDF1_TF;
```

```
/**/
HVPUDF1_SCALAR:PROCEDURE RETURNS(BIN FIXED(31));
 DCL MYRC BIN FIXED(31) INIT(Ø);
 DCL TELLER BIN FIXED(31) BASED(ADDR(UDF_SPAD_TEXT));
 IF TELLER>2 THEN DO;
  /*PREVENT ENDLESS LOOP WHEN SCALAR CALLED AS TABLE FUNCTION*/
  MYRC=386Ø2;
  UDF_SQLSTATE='386Ø2';
  DCL TELLER_PIC PIC '99999';
  TELLER_PIC=TELLER;
  UDF_DIAG_MSG='UNEXPECTED TELLER '||TELLER_PIC;
  GOTO HVPUDF1_SCALAR_EXIT;
 END;
 TELLER=TELLER+1;
 IF UDF_CALL_TYPE=$CALL_FIRST_SCALAR THEN DO;
  LEGISJ1=PJ1;
  LEGISJ2=PJ2;
  ZITTING=PZ;
  NR     =PNR;
  EXEC SQL SELECT
              LEGMVJ1,
              LEGMVJ2,
              ZITNGMV,
              INTERMV,
              M9TXTN11,
              M9TXTN12 ,
              M9TXTN13 ,
              M9TXTN14 ,
              M9TXTN15 ,
              M9TXTN16 ,
              M9TXTN17 ,
              M9TXTN18 ,
              M9TXTN19 ,
              M9TXTF11 ,
              M9TXTF12 ,
              M9TXTF13 ,
              M9TXTF14 ,
              M9TXTF15 ,
              M9TXTF16 ,
              M9TXTF17 ,
              M9TXTF18 ,
              M9TXTF19
        INTO :APMVRØ1
        FROM XXXX.YY
        WHERE LEGMVJ1 = :LEGISJ1
          AND LEGMVJ2 = :LEGISJ2
          AND ZITNGMV = :ZITTING
          AND INTERMV = :NR;
   MYRC=SQLCODE_CHECK('SELECT');
   /*SIGNAL ERROR*/
   IF MYRC¬=Ø THEN DO;
```

```
    UDF_SQLSTATE='386Ø2';
    UDF_DIAG_MSG='UNEXPECTED SQLCODE '||SQLCODE_PIC;
   END;
   ELSE DO;
    UDF_INDR=Ø;
    IF PLANG='B' THEN DO;
     UDF_RESULT=TEXTN()||
       '@@@@'||
       TEXTF();
    END;/*B*/
    ELSE IF PLANG='N' THEN DO;
     UDF_RESULT=TEXTN();
    END;/*N*/
    ELSE  DO;
     UDF_RESULT=TEXTF();
    END;/*F*/
   END;/*MYRC=Ø*/
 END;/*$CALL_FIRST_SCALAR*/
 HVPUDF1_SCALAR_EXIT:
 RETURN(MYRC);
END HVPUDF1_SCALAR;
/**/
TEXTN:PROCEDURE RETURNS(CHAR(675) VARYING);
 DCL RV CHAR(675) VARYING INIT('');
 /*
 NOG DOEN||
 IF M9TXTN12='' THEN FLAG=$TRUE
 RV=RV||RTRIM(M9TXTN11,FLAG)
 */
 RV=   RTRIM(M9TXTN11,$FALSE)||
       RTRIM(M9TXTN12,$FALSE)||
       RTRIM(M9TXTN13,$FALSE)||
       RTRIM(M9TXTN14,$FALSE)||
       RTRIM(M9TXTN15,$FALSE)||
       RTRIM(M9TXTN16,$FALSE)||
       RTRIM(M9TXTN17,$FALSE)||
       RTRIM(M9TXTN18,$FALSE)||
       RTRIM(M9TXTN19,$FALSE);
 RETURN(RV);
END TEXTN;
TEXTF:PROCEDURE RETURNS(CHAR(675) VARYING);
 DCL RV CHAR(675) VARYING INIT('');
 RV=   RTRIM(M9TXTF11,$FALSE)||
       RTRIM(M9TXTF12,$FALSE)||
       RTRIM(M9TXTF13,$FALSE)||
       RTRIM(M9TXTF14,$FALSE)||
       RTRIM(M9TXTF15,$FALSE)||
       RTRIM(M9TXTF16,$FALSE)||
       RTRIM(M9TXTF17,$FALSE)||
       RTRIM(M9TXTF18,$FALSE)||
       RTRIM(M9TXTF19,$FALSE);
```

```
  RETURN(RV);
 END TEXTF;
 /*herman vierendeels systems programmer belgium*/
 END HVPUDF1;
```

HVPUDF1 DB2 DEFINITIONS

```
//DB2DEF JOB 'FNCTNS',CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//FNCTNS EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSNT)
 RUN PROGRAM(DSNTIAD)  PLAN(DSNTIA61) -
    LIB('DSN610.RUNLIB.LOAD')
 END
/*
//SYSPRINT DD  SYSOUT=X
//SYSIN    DD  *
  CREATE FUNCTION TSHVR.HVPUDF2(
  VARCHAR(4096))
  RETURNS
     TABLE(
       FIELD1        VARCHAR(4096))
  SPECIFIC  TSHVR.HVPUDF2
  EXTERNAL NAME 'HVPUDF1'
  LANGUAGE PLI
  PARAMETER STYLE DB2SQL
  PROGRAM TYPE MAIN
  DETERMINISTIC
  READS SQL DATA
  NO EXTERNAL ACTION
  SCRATCHPAD 100
  FINAL CALL
  DBINFO
  WLM ENVIRONMENT DBPRCRX1
  COLLID TSHVRT
  STAY RESIDENT NO;

  CREATE FUNCTION TSHVR.HVPUDF1(VARCHAR(4096))
  RETURNS VARCHAR(4096)
  SPECIFIC  TSHVR.HVPUDF1
  EXTERNAL NAME 'HVPUDF1'
  LANGUAGE PLI
  PARAMETER STYLE DB2SQL
  PROGRAM TYPE MAIN
  DETERMINISTIC
  READS SQL DATA
  NO EXTERNAL ACTION
  SCRATCHPAD 100
  FINAL CALL
  DBINFO
```

```
  WLM ENVIRONMENT DBPRCRX1
  COLLID TSHVRT
  STAY RESIDENT NO;
  COMMIT;
/*
//
```

HVPUDF1 JCL

```
//DBPRCRX1 PROC DB2SSN=DSNT,NUMTCB=2,APPLENV=DBPRCRX1
//*
//DBPRCRX1 EXEC PGM=DSNX9WLM,TIME=1440,
//              PARM='&DB2SSN,&NUMTCB,&APPLENV',
//              REGION=0M
//STEPLIB  DD  DISP=SHR,DSN=DSN610.RUNLIB.LOAD
//         DD  DISP=SHR,DSN=SYS8.CEE8.SCEERUN
//         DD  DISP=SHR,DSN=DSN610.SDSNLOAD
//SYSTSPRT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  DUMMY
//DSNTRACE DD  SYSOUT=*
```

Herman Vierendeels
Systems Programmer (Belgium)

Monitoring DataPropagator on MVS – part 2

This month we conclude the code to monitor the propagation process from DB2 on one MVS system to DB2 on a different MVS system.

BIND DPRMON

Check that you have established connectivity between the DB2 systems. If the remote DB2 is not on MVS, I believe you will need DB2 Connect and DataJoiner for this. Then run the following in a batch TSO job step, on the MVS system on which the monitor will run.

```
  DSN SYSTEM(local-ssid)
  BIND PACKAGE(local-location"."collection-id) MEMBER(DPRMON) -
    ACTION(REPLACE) DBPROTOCOL(DRDA) -
    ISOLATION(UR) OWNER(owner) CURRENTDATA(NO) -
    RELEASE(COMMIT) VALIDATE(BIND)
  BIND PACKAGE(remote-location"."collection-id) MEMBER(DPRMON) -
```

```
    ACTION(REPLACE) DBPROTOCOL(DRDA) -
    ISOLATION(UR) OWNER(owner) CURRENTDATA(NO) -
    RELEASE(COMMIT) VALIDATE(BIND)
  BIND PLAN(DPRMON) -
    ACTION(REPLACE) RETAIN DBPROTOCOL(DRDA) -
    ISOLATION(UR) OWNER(owner) CURRENTDATA(NO) -
    ACQUIRE(USE) RELEASE(COMMIT) VALIDATE(BIND) -
    PKLIST("*."collection-id".*")
  END
```

Note this example assumes that ‘owner’ is the authid that the monitor will use and that it is authorized on both the local and remote DB2 subsystems. If that is not the case, you may be able to define the AUTHID and NEWAUTHID columns in the Communications Data Base table, SYSIBM.USERNAMES, to get around that (assuming ZPARM TCPALVER=YES).

ASSEMBLE AND LINK ASSEMBLER PROGRAM WAITA

```
         TITLE 'WAIT FOR THE USER-SPECIFIED TIME'
*---------------------*  FUNCTION  *-------------------------------*
* WAITA                                                            *
* _____                                                            *
* To wait for the specified time period, or until user presses ATTN.  *
*                                                                  *
* It sets a long timer with WAIT=NO for the user specified period.    *
* It also sets a timer with WAIT=YES, and after that wait it checks   *
* whether ATTN has been pressed.  If no ATTN: a new 1/2 second wait   *
* is started.  This repeats until either the long timer expires or    *
* ATTN is pressed.                                                 *
*                                                                  *
* Usage in TSO     : CALL 'load-library(WAITA)' 'hhmmssth'         *
* Usage in TSO/ISPF: ISPEXEC SELECT PGM(WAITA) PARM(hhmmssth)      *
* Usage in JCL     : //stepname EXEC PGM=WAITA,PARM=hhmmssth       *
*------------------------------------------------------------------*
*  RETURN CODE : Ø  = TIMER EXPIRED                                *
*                4  = ATTN KEY PRESSED BY TSO USER                 *
*                8  = BAD PARM LENGTH                              *
*                12 = NO PARMS SUPPLIED                            *
*                16 = BAD PARM SYNTAX                              *
*                2Ø = STIMER ERROR                                 *
********************************************************************
*  REGISTER USAGE
R1       EQU   1           PARM ADDRESS
R2       EQU   2           WORK
R3       EQU   3           WORK
BASEREG  EQU   11          PROGRAM BASE
R12      EQU   12          ATTN BASE
```

```
R15      EQU   15          RETURN CODE
         EJECT
*********************************************************************
WAITA    RMODE ANY
WAITA    AMODE 31
WAITA    CSECT
         USING WAITA,R15                SET UP ADDRESSABILITY
         SAVE  (14,12)
         LR    BASEREG,R15
         DROP  R15
         USING WAITA,BASEREG
         B     START                    GO AROUND EYECATCHER
PROGNAME DC    CL8'WAITA'               EYECATCHER
         DC    CL8' VERSION'
VERSION  DC    CL6' Ø1.Ø1'
DATE_ASM DC    CL11' &SYSDATE'          DATE ASSEMBLED
TIME_ASM DC    CL8' &SYSTIME'           TIME ASSEMBLED
*----------------- PROCESS PARM ----------------------------------*
START    DS    ØH
         LTR   R1,R1                    ANY PARMS ?
         BZ    NOPARMS                  NO, EXIT RC=12
         ICM   R1,B'1111',Ø(R1)         -> PARMS
         BZ    NOPARMS                  IF NO PARMS, EXIT RC=12
         CLC   Ø(2,R1),=AL2(L'WAIT_TIME) GOOD LENGTH ?
         BNE   BADLEN                    NO, EXIT RC=8
         LA    R2,2(R1)                 -> PARM
         LA    R3,L'WAIT_TIME           R3 = LENGTH OF PARM
LOOP     DS    ØH                       CHECK ALL CHARS
         CLI   Ø(R2),C'Ø'               LT Ø ?
         BL    BADPARM                  YES, EXIT RC=16
         CLI   Ø(R2),C'9'               GT 9 ?
         BH    BADPARM                  YES, EXIT RC=16
         LA    R2,1(R2)
         BCT   R3,LOOP
         MVC   WAIT_TIME,2(R1)          SET STIMERM PARM
         MVC   FIRST_WAIT+6(2),WAIT_TH  SET FIRST WAIT TIME
*----------------- EXIT IF ATTN PRESSED --------------------------*
SETATTN  DS    ØH
         STAX  ATTNEXIT,REPLACE=YES,DEFER=NO,TOPLEVL=YES
*----------------- CALL STIMERM FOR LONG 'WAIT' ------------------*
LONGWAIT DS    ØH
         MVI   EXPIRE,C'N'              SET EXPIRE = NO
         STIMERM SET,WAIT=NO,DINTVL=WAIT_TIME,ID=MYID,                X
               EXIT=LONGEND,ERRET=BADTIME
*
*------------------ WAIT FOR A FRACTION OF A SECOND ---------------*
FIRSTWAIT DS   ØH
         STIMER WAIT,DINTVL=FIRST_WAIT,ERRET=ATTNEND
*----------------- LOOP THRU 1/2 SEC STIMER WAITS ----------------*
WAITLOOP DS   ØH
         CLI   ATTN,C'Y'                WAS ATTN ROUTINE ENTERED?
```

```
         BE    ATTNEND                  YES, EXIT RC=4
         CLI   EXPIRE,C'Y'              HAS LONG TIMER EXPIRED?
         BE    EXIT                     YES, EXIT RC=Ø
         STIMER WAIT,DINTVL=WAITABIT,   WAIT FOR ANOTHER HALF SECOND  X
               ERRET=ATTNEND
         B     WAITLOOP                 THAT WAS FUN,LET'S DO IT AGAIN
EXIT     DS    ØH
         RETURN (14,12),RC=(15)         RETURN TO CALLER
         EJECT
*********************************************************************
* SETTING (NON-ZERO) RETURN CODES FOR EXIT                           *
*********************************************************************
ATTNEND  DS    ØH                       ATTENTION PRESSED
         LA    R15,4                    SET RETURN CODE
         B     EXIT                     RETURN TO CALLER
BADLEN   DS    ØH                       BAD PARM LENGTH
         LA    R15,8                    SET BAD RETURN CODE
         B     EXIT                     RETURN TO CALLER
NOPARMS  DS    ØH                       NO PARMS SUPPLIED
         LA    R15,12                   SET BAD RETURN CODE
         B     EXIT                     RETURN TO CALLER
BADPARM  DS    ØH                       BAD SYNTAX
         LA    R15,16                   SET BAD RETURN CODE
         B     EXIT                     RETURN TO CALLER
BADTIME  DS    ØH                       PROBLEM WITH STIMERM CALL
         LA    R15,2Ø                   SET BAD RETURN CODE
         B     EXIT                     RETURN TO CALLER
         EJECT
*********************************************************************
* TIMER EXIT (STIMERM MACRO)                                         *
*********************************************************************
LONGEND  DS    ØD
         SAVE  (14,12)
         BALR  R12,Ø
         USING *,R12
         MVI   EXPIRE,C'Y'              INDICATE TIMER HAS EXPIRED
         RETURN (14,12)
*********************************************************************
* STAX EXIT(ATTENTION ROUTINE)                                       *
*********************************************************************
ATTNEXIT DS    ØH
         SAVE  (14,12)
         BALR  R12,Ø
         USING *,R12
         STAX  ATTNEXIT,REPLACE=NO
         MVI   ATTN,C'Y'               INDICATE ATTN WAS PRESSED
         STIMERM CANCEL,ID=ALL         CANCEL ALL STIMERM REQUESTS
         RETURN (14,12),RC=Ø
*********************************************************************
CONSTNTS DS    ØF
         LTORG ,
```

```
          DS  ØD
WAIT_TIME  DS  ØCL8
WAIT_HRS   DC  CL2'ØØ'                  hours to wait
WAIT_MIN   DC  CL2'ØØ'                  minutes to wait
WAIT_SEC   DC  CL2'ØØ'                  seconds to wait
WAIT_TH    DC  CL2'ØØ'                  tenths/hundredths to wait
FIRST_WAIT DC  CL8'ØØØØØØØØ'            PARM for first wait time
WAITABIT   DC  CL8'ØØØØØØ5Ø'            PARM to wait 1/2 second
MYID       DC  CL4'ZZZZ'                Timer Id
EXPIRE     DC  CL1'N'                   has LONG Timer EXPIRED?
ATTN       DC  CL1'N'                   was ATTENTION pressed?
         END   WAITA
```

REXX EXEC DPROPMON

Copy into a library.

```
 /*============================  REXX  ================================*/
 /* DpropMon:  Monitor for DataPropagator Apply & Capture             */
 /*                                                                   */
 /* This is invoked in a batch job.  It produces a report of the lag  */
 /* times in propagation.  By default it checks every 5 minutes, but  */
 /* that can be altered by a start parameter.                         */
 /*                                                                   */
 /* Program DPRMON is called to get data from Dprop DB2 tables.       */
 /* Program WAITA is called to wait for a specified time.             */
 /*                                                                   */
 /* Written by Ron Brown       Version 1.13          August 2ØØ2      */
/*===================================================================*/
   Address TSO                          /* commands go to TSO        */
   warn = 'DpropMon: Warning -'         /* start of warning msg text */
   abnd = 'DpropMon: Abending -'        /* start of abend msg text   */
   /*---------------------*/
   /* get start parameters */
   /*---------------------*/
   Arg parms
   Parse Value parms With ,
      1 'SSID('   ssid          ')',
      1 'QUAL('   apply_qual    ')',
      1 'SUBSET(' sub_set       ')',
      1 'APPLY('  apply_task    ')',
      1 'CAPT('   capt_task     ')',
      1 'WAIT('   wait_time     ')',
      1 'MAXLAG(' max_synch_lag ')',
      1 'MAXC('   max_cycle_lag ')',
      1 'MSGS('   send_msgs     ')',
      1 'NOTIFY(' userids       ')',
      1 'END('    endtime       ')'
   /*-----------------------------------------------------*/
   /* check parms and set defaults if no parameter supplied */
```

```
/*-------------------------------------------------------*/
Call CHECK_PARMS
If result > Ø Then Exit result      /* result = 16 if error(s) */
no_capt  = Ø
no_apply = Ø
capt_lag = -1
synch_lag = -1
cycle_lag = -1
last_timestamp = '??????????????????????????'
/*---------------------*/
/* Main processing loop */
/*---------------------*/
Do mon_cycle_no = 1 To 99999999    /* ie forever */
   last_curr_time = curr_time           /* save previous values .. */
   last_capt_lag = capt_lag             /* ..to be able to compare */
   last_synch_lag = synch_lag
   last_cycle_lag = cycle_lag
   waita_time = Time('R')               /* time spent in WAITA pgm */
   /*-----------------------*/
   /* check the time to stop */
   /*-----------------------*/
   If endtime <> '' Then Do        /* if user specified an endtime */
      Call CHECK_ENDTIME
      If result > Ø Then Exit Ø              /* Exit */
      End
   /*---------------------------------*/
   /* run the DPRMON program under DSN */
   /*---------------------------------*/
   Newstack                 /* start a new (empty) stack */
   Push ''
   Push 'END'
   dprmon_parms = last_timestamp wait_time apply_qual sub_set
   Push "RUN PROGRAM(DPRMON) PLAN(DPRMON)" "PARMS('"dprmon_parms"')"
   "DSN SYSTEM("ssid")"   /* invoke DSN pgm using input from stack */
   If rc > Ø Then Do
      Call PGM_ERROR_MSGS('DPRMON' rc dprmon_parms)
      Exit 16               /* Exit if any program error */
      End
   Delstack                 /* finished with that stack */
   /*---------------------------------*/
   /* read the output from DPRMON program */
   /*---------------------------------*/
   "EXECIO * DISKR DPRWORK (STEM line. FINIS"
   Parse Var line.1 sqlcode_c capt_loc sqlcode_a cntl_loc ,
     capt_location cntl_location targ_location last_timestamp ,
     time_diff .
   Parse Var line.2 curr_time capt_lag set_name synch_date,
     synch_time synch_lag cycle_lag apply_active commit_int,
     sleep_min max_synch_min prune_int cycles changes .
   If curr_time = '??.??.??' Then curr_time = Time()
   /*-----------------------------------------*/
```

```
/* write the headings at the top of the report */
/*-------------------------------------------*/
If mon_cycle_no = 1 Then Do  /* if this is first monitor cycle */
   Call START_REPORT
   If result > 0 Then Exit result   /* result = 16 if error(s) */
   End
/*-----------------------------------------------------------*/
/* check values from DB2, create comments or msgs for problems */
/*-----------------------------------------------------------*/
Select
When Pos('1900',synch_date) > 0 Then Do       /* '1900-01-01' */
   warn_stars = '***'
   cycle_lag = '?????'
   synch_lag = '?????'
   comment = 'invalid Last Synchpoint value, lag unknown'
   End
/*-------------------------------------------*/
/* if lag is too long - send warning messages */
/*-------------------------------------------*/
When synch_lag > max_synch_lag Then Do
   /*-----------------------------------*/
   /* check that Apply/Capture is running */
   /*-----------------------------------*/
   If capt_loc = 'L' &,         /* if Capture runs on this MVS */
      capt_task <> '' Then
      no_capt = CHECK_TASK(capt_task)
   If apply_task <> '' Then       /* if Apply runs on this MVS */
      no_apply = CHECK_TASK(apply_task)
   /*-----------------------------------------------------*/
   /* select message to send & warning comment for report */
   /*-----------------------------------------------------*/
   Call SELECT_WARNING     /* sets variables: msg & comment */
   /*-------------------------------*/
   /* send message to master console */
   /*-------------------------------*/
   If send_msgs = 'CONSOLE' | send_msgs = 'BOTH' Then
      "SEND '"msg"     '"
   /*----------------------------*/
   /* send message to TSO user(s) */
   /*----------------------------*/
   If (send_msgs = 'TSO' | send_msgs = 'BOTH'),
    & userids <> '' Then
      "SEND '"Time() Date('E') ' ' msg"  ' USER("userids") LOGON"
   End
Otherwise              /* lag was not too long ......    */
   comment = ''        /* no warning text when lag is OK */
   warn_stars = '   '  /* no highlight when lag is OK    */
   End
/*---------------------------------------*/
/* write a new line of data in the report */
/*---------------------------------------*/
```

```
      Say warn_stars curr_time Right(capt_lag,8) Right(cycles,8),
        Right(changes,7) Right(cycle_lag,7) Right(synch_lag,9),
        Right(synch_time,10) Right(synch_date,11) '  ' comment
      Say
      /*-----------------------------------*/
      /* write SQL error messages in report */
      /*-----------------------------------*/
      If sqlcode_c <> 0 | (sqlcode_a <> 0 & sqlcode_a <> 100) Then
         Call WRITE_ERRMSGS
      /*-------------------------------------------------------------*/
      /* wait for the specified time before the next monitor cycle */
      /*-------------------------------------------------------------*/
      db2_time = Time('R') /* time taken to get & process DB2 data */
      Call CALC_WAIT_PARM         /* calculate how long to wait */
      "CALL *(WAITA) '"wait_parm"'"     /* external program */
      If rc <> 0 Then Do
         Call PGM_ERROR_MSGS('WAITA' rc wait_parm)
         Exit 16               /* Exit if any program error */
         End
      End        /* end of main processing loop */
   Say
   Exit 0        /* That's all folks! */
/*====================================================================*/
/* check parms and set defaults if no parameter supplied              */
/*--------------------------------------------------------------------*/
CHECK_PARMS:
   check_code = 0
   If ssid = '' Then Do
      "SEND '"abnd ,
          "SSID parameter MUST be specified            '"
      check_code = 16
      End
   If apply_qual = '' Then Do
      apply_qual = ssid        /* DJRA default is to use alias name */
      "SEND 'DpropMon:",
          "QUAL parameter not specified - assumed to be:" ssid "    '"
      End
   If sub_set = '' Then Do
      sub_set = ssid        /* DJRA default is to use alias name */
      "SEND 'DpropMon:",
          "SUBSET parameter not specified - assumed to be:" ssid "    '"
      End
   warn = warn apply_qual '/' sub_set '-'
   abnd = abnd apply_qual '/' sub_set '-'
   /* apply_task to be specified only when it runs on same MVS system */
   If apply_task = '' Then NOP
   /* capt_task to be specified only when it runs on same MVS system  */
   If capt_task = '' Then NOP

   If max_synch_lag = '' Then Do
```

```
   max_synch_lag = 300
   "SEND 'DpropMon:",
       "MAXLAG parameter not specified - set to 300 seconds        '"
   End
If max_synch_lag = 0 | Datatype(max_synch_lag) = 'CHAR' Then Do
   "SEND '"abnd ,
       "Invalid parameter MAXLAG("max_synch_lag"), must be numeric '"
   check_code = 16
   End
If max_cycle_lag = '' Then
   max_cycle_lag = max_synch_lag - 3
If max_cycle_lag = 0 | Datatype(max_cycle_lag) = 'CHAR' Then Do
   "SEND '"abnd ,
       "Invalid parameter MAXC("max_cycle_lag"), must be numeric   '"
   check_code = 16
   End
If wait_time = '' Then Do
   wait_time = '5'     /* set default:  ie 5 minutes */
   "SEND 'DpropMon:",
       "WAIT parameter not specified - set to 5 minutes            '"
   End
If wait_time <= 0 | Datatype(wait_time) = 'CHAR',
 | Length(wait_time) = 3 ,
 | Length(wait_time) > 4 Then Do
   "SEND '"abnd ,
       "Invalid parameter WAIT("wait_time"), 1 - 99 is valid      '"
   Return 16
   End
wait_m = Format(Left(wait_time,2))      /* minutes */
If Length(wait_time) = 4 Then Do           /* format = 'mmss' */
   wait_s = Format(Right(wait_time,2))     /* seconds */
   wait_secs = (60 * wait_m) + wait_s      /* converted to secs */
   If wait_s = 0        /* create text for report */
      Then wait_text = ''
      Else wait_text = wait_s 'secs'
   If wait_m > 0
      Then wait_text = wait_m 'mins' wait_text
   End
Else Do    /* Length(wait_time) = 1 or 2 */
   wait_s = '00'
   wait_secs = 60 * wait_time              /* converted to secs */
   wait_text = Format(wait_time) 'mins'
   wait_time = Right(wait_time,2,'0')'00'  /* convert to 'mmss' */
   End
Select
  When Pos(send_msgs,'TSO') = 1 Then   /* match at least 1st char */
       send_msgs = 'TSO'               /* only to TSO userids */
  When Pos(send_msgs,'CONSOLE') = 1 Then
       send_msgs = 'CONSOLE'           /* only to system console */
  When Pos(send_msgs,'NONE') = 1 Then
```

```
        send_msgs = 'NONE'              /* send no messages */
    Otherwise                           /* default is 'BOTH' */
        send_msgs = 'BOTH'              /* to TSO & system console */
    End
  Return check_code        /* Ø = OK,   16 = error(s)  */
/*================================================================*/
/* write the headings at the top of the report                    */
/*----------------------------------------------------------------*/
START_REPORT:
  sqlcode_c = Format(sqlcode_c)     /* remove leading zeros */
  sqlcode_a = Format(sqlcode_a)
  If sqlcode_c = 1ØØ & sqlcode_a = 1ØØ Then Do
     Say 'Unable to find that Subscription Set in the DataPropagator',
         'Control tables'
     Say
     "SEND '"abnd ,
         "Unable to find Subscription Set:" sub_set "    '"
     Return 16
     End
  Say
  Say
  Say
  Say '       DATAPROPAGATOR MONITOR             (Version 1.13)'
  Say '     ================================================='
  Say
  Say '       Apply Qualifier :' apply_qual
  Say '       Subscription Set:' set_name
  Say
  Say 'This monitor is checking DataPropagator Control tables every',
  wait_text'.'
  Say
  If synch_time = '??.??.??' Then Do
     Say 'The monitor was unable to read data from the Apply',
         'Subscription Set table.'
     Say 'Therefore, the following information is not complete:'
     End
  If (synch_time <> '??.??.??' & synch_date <> 'Ø1.Ø1.19ØØ' &,
      1Ø + (6Ø * sleep_min) > wait_secs) Then Do
     Say 'That is faster than the Apply cycles, and thus less',
         'than recommended:'
     Say 'With every monitor cycle it checks if propagation is too',
         'far behind'
     Say '(ie more than the MAXLAG).  If so, it usually checks',
         'whether a new'
     Say 'Apply cycle has been started since the last monitor cycle,',
         'to check'
     Say 'if Apply is started but not actually working.       '
     Say 'But the monitor cycle time is too short, so that check is',
         'not done.'
     Say
```

```
   wait_rec = sleep_min + 1
   "SEND 'DpropMon:",
       "WAIT time too short - at least" wait_rec "recommended      '"
   End
If (sqlcode_c = 0 & 2 * commit_int > wait_secs) Then Do
   Say 'Note: The monitor cycle time is less than 2 Capture commit',
       'intervals,'
   Say 'hence it cannot detect when Capture is not reading the logs.'
   Say
   End
Say
If sqlcode_a = 0 Then Do
   Say 'Apply running with SLEEP_MINUTES :' sleep_min
   Say 'Apply cycle MAX_SYNCH_MINUTES ...:' max_synch_min
   End
If sqlcode_c = 0 Then Do
   Say 'Capture runs with COMMIT_INTERVAL:' commit_int
   Say 'CD & UOW tables PRUNE_INTERVAL ..:' prune_int
   End
Say
mvs = MVSVar('SYSNAME')
Say 'Monitor is running on MVS system :' mvs
Say 'Monitor connected to DB2 ssid ...:' ssid
If capt_loc = 'L'
  Then Say 'Source tables at local DB2 ......:' capt_location
  Else Say 'Source tables at remote location.:' capt_location
If cntl_loc = 'L'
  Then Say 'Control tables at local DB2 .....:' cntl_location
  Else Say 'Control tables at remote location:' cntl_location
Select
  When targ_location = cntl_location Then targ_loc = cntl_loc
  When targ_location = capt_location Then targ_loc = capt_loc
  Otherwise targ_loc = 'R'
  End
If targ_loc = 'L'
  Then Say 'Target tables at local DB2 ......:' targ_location
  Else Say 'Target tables at remote location.:' targ_location
If apply_task <> '' Then Do        /* if Apply runs on this MVS */
   no_apply = CHECK_TASK(apply_task)
   If smfid = 'SMFID' | smfid = '' Then smfid = mvs
   Say 'Apply task runs on' smfid 'system ..:' apply_task
   End
If capt_task <> '' & capt_loc = 'L' Then Do
   no_capt = CHECK_TASK(capt_task)
   If smfid = 'SMFID' | smfid = '' Then smfid = mvs
   Say 'Capture task runs on' smfid 'system :' capt_task
   End
If send_msgs = 'NONE' Then
   Say 'No warning messages will be sent .'
If send_msgs = 'CONSOLE' | send_msgs = 'TSO' Then
```

```
   Say 'Warning messages sent only to .. :' send_msgs
If (send_msgs = 'BOTH' | send_msgs = 'TSO'),
 & userids <> '' Then
   Say 'TSO Userids to notify if problems:' userids
If endtime <> '' Then
   Say 'This monitor will stop running at:' endtime
Say
Say
Say 'Report columns'
Say '--------------'
If cntl_loc = 'R'
   Then lcntl = 'remote DB2' cntl_location
   Else lcntl = 'the local MVS system'
Say ' Time ..........: Current time at' lcntl'.'
Say '                  "***" is shown before the time if there is',
                       'a warning message.'
If sqlcode_c = Ø
   Then cl = Strip(Format(commit_int * 1.5,5,Ø))
   Else cl = '????'
Say ' Capture Lag ...: How long Capture lags behind current time',
                        'reading DB2 logs.'
Say '                   Normally in the range 2 -' cl 'seconds.'
Say ' Apply Cycles ..: Number of successful Apply subscription',
                        'cycles during the'
Say '                   last monitor cycle (ie. in the last',
                        wait_text').'
Say ' Apply Changes .: Total number of INSERTS, UPDATES & DELETES',
                        'done by Apply'
Say '                   in the last' wait_text'.'
Say ' Apply Lag .....: How long the last Apply cycle was',
                        'behind when it started,'
If time_diff <> Ø Then Do
   Say '                  (allowing for the time difference between',
                           'the DB2'
   Say '                   sub-systems, which is currently',
                           Abs(time_diff) 'seconds).'
   End
If sqlcode_c = Ø
   Then alagmax = cl + 5
   Else alagmax = '????'
Say '                  Normally in the range 3 -' alagmax 'seconds.'
Say ' Synchpoint Lag.: How long the last change in the target',
                        'tables lags behind'
If time_diff = Ø Then
   Say '                    the current time.'
If time_diff <> Ø Then Do
   Say '                   the current time, (allowing for DB2 time',
                           'differences).'
   End
If sqlcode_a = Ø & sqlcode_c = Ø
```

```
     Then slagmax = (sleep_min * 60) + cl+10
     Else slagmax =  '????'
  Say '                   Normally in the range 5 -' slagmax 'seconds.'
  Say ' Last Synchpoint: Time & date of the last change applied to',
                         'the target tables,'
  Say '                   by the last successfully completed Apply',
                         'cycle.'
  If time_diff <> 0 Then Do
     If capt_loc = 'R'
        Then lcapt = 'remote DB2' capt_location
        Else lcapt = 'the local MVS system'
     If time_diff > 0
        Then td = 'ahead of'
        Else td = 'behind'
     Say '                    Note that this time is from' lcapt',',
                             'which has'
     Say '                    a current time' Abs(time_diff) 'seconds',
                              td lcntl'.'
     End
  Say ' Warning .......: Problem description, if propagation is',
                         'too far behind.'
  Say '                   That is when Synchpoint Lag >',
                          max_synch_lag 'seconds.'
  Say
  Say
  Say '               Capture     ---- Apply ----         --',
      'Last Synchpoint --'
  Say '      Time       Lag     Cycles Changes   Lag       Lag   ',
      '  Time       Date              Warning'
  Say '    --------   ------   ------ ------ ------   ------ ',
      '--------  ----------    --------------------------------'
  Return 0
/*====================================================================*/
/* check whether it is time to stop this monitor yet                  */
/*--------------------------------------------------------------------*/
CHECK_ENDTIME:
  If mon_cycle_no = 1 Then Do
     If Time() > endtime Then reltime = 'HIGH'
     End
  Else Do
     If reltime = 'HIGH' & Time() < endtime Then
        reltime = 'LOW'
     If reltime <> 'HIGH' & Time() > endtime Then Do
        Say "   *** It's getting late now, so I'll stop.  Goodbye! ***"
        Return 16            /* Exit */
        End
     End
  Return 0
/*====================================================================*/
/* check that a started task is running on this MVS system            */
```

```
/*-----------------------------------------------------------------*/
CHECK_TASK:
  Arg task
  x = OutTrap('stat_msg.',500)   /* up to 500 lines output */
  Address TSO "STATUS" task /* issue TSO STATUS cmd  */
  x = OutTrap('Off')
  no_task = 1
  Do i = stat_msg.0 To 1 By -1
     If Pos('IAT8968',stat_msg.i) > 0 Then Do    /* JES3 message */
        Parse Var stat_msg.i mnun jb jnam jnum snam ' ON ' smfid .
        If smfid <> '' Then Do
           no_task = 0
           Leave i
           End
        End
     If Pos('EXECUTING',stat_msg.i) > 0 Then Do
        no_task = 0
        Leave i
        End
     End
  Return no_task
/*=================================================================*/
/* select message to send & warning comment for report             */
/*-----------------------------------------------------------------*/
SELECT_WARNING:
  warn_stars = '***'
  /* get complete monitor cycle loop time in sec */
  If mon_cycle_no = 1 Then
     loop_secs = wait_secs
  Else Do /* loop_secs = difference between last two DB2 times */
     last_curr_hr = Left(last_curr_time,2)
     last_curr_min = Substr(last_curr_time,4,2)
     last_curr_sec = Right(last_curr_time,2)
     curr_hr = Left(curr_time,2)
     curr_min = Substr(curr_time,4,2)
     curr_sec = Right(curr_time,2)
     loop_secs = (3600 * curr_hr) + (60 * curr_min) + curr_sec - ,
       (3600 * last_curr_hr) - (60 * last_curr_min) - last_curr_sec
     End
  Select
    When (sqlcode_c = 0 & capt_loc = 'L' & no_capt) Then
       comment = 'Capture ('capt_task') not started on' mvs '  '
    When (sqlcode_c = 0 & mon_cycle_no <> 1 &,
          capt_lag - last_capt_lag + 1 >= loop_secs &,
          wait_secs >= 2 * commit_int) Then
       comment = 'Capture not reading logs             '
    When no_apply Then
       comment = 'Apply ('apply_task') not started on' mvs '  '
    When (sqlcode_a = 0 & apply_active = 0) Then
       comment = 'This Subscription Set is deactivated  '
```

```
    When (sqlcode_a = Ø & cycles = Ø &,
          1Ø + (6Ø * sleep_min) < loop_secs &,
          synch_lag - last_synch_lag >= loop_secs &,
          last_cycle_lag = cycle_lag) Then
       comment = 'Apply not working (no new cycle)      '
    When (sqlcode_a = Ø & max_cycle_lag > cycle_lag ) Then
       comment = 'Lag >' max_synch_lag 'secs, check Apply           '
    When sqlcode_a = Ø Then
       comment = 'Lag >' max_synch_lag 'secs,',
             'check Capture & Apply  '
    When sqlcode_a <> Ø & (sqlcode_c = Ø | sqlcode_c = -99999) Then
       comment = 'error reading Apply table,',
             'sqlcode =' sqlcode_a '   '
    When (sqlcode_a = Ø & sqlcode_c <> Ø) Then
       comment = 'error reading Capture table,',
             'sqlcode =' sqlcode_c '   '
    When (sqlcode_a <> Ø & sqlcode_c <> Ø) Then
       comment = 'error reading Apply & Capture tables,',
             'sqlcodes =' sqlcode_a sqlcode_c '  '
    Otherwise
    End
  msg = warn comment
  Return
/*================================================================*/
/* write SQL error messages in report                             */
/*----------------------------------------------------------------*/
WRITE_ERRMSGS:
  If sqlcode_a <> Ø Then
     Say "PROGRAM(DPRMON) PARMS('"dprmon_parms"')"
  /* write diagnostic messages  (formatted by DSNTIAR) */
  Do i = 3 To line.Ø
     If line.i <> ' ' | last_line = 'NONBLANK' Then
        Say Right(line.i,131)
     If line.i = ' '
        Then last_line = 'BLANK'
        Else last_line = 'NONBLANK'
     End
  Say
  Say
  Return
/*================================================================*/
/* calculate value for WAIT program parm from time spent in DB2   */
/*----------------------------------------------------------------*/
CALC_WAIT_PARM:
  /* set wait_parm = wait_time - db2_time - MVS_dispatch_time */
  wait_parm = (wait_m * 6Ø) + wait_s - db2_time  /* in sec */
  wait_parm = Trunc((wait_parm * 1ØØ) - 1Ø) /* Ø.1Ø sec to dispatch */
  wait_parm = Right(wait_parm % 6ØØØ,4,'Ø') ||,       /* 'hhmm' */
              Right(wait_parm // 6ØØØ,4,'Ø')          /* 'ssth' */
  If Datatype(wait_parm) = 'CHAR'  /* ie. when '-' char in string! */
```

```
     Then wait_parm = '00000000'
  Return
/*====================================================================*/
/* error messages when DPRMON or WAITA programs fail                  */
/*--------------------------------------------------------------------*/
PGM_ERROR_MSGS:
  Arg pgmname retcode pgm_parms
  Say
  Say
  Say '***' pgmname 'ended with RC =' retcode ' *** ABENDING ***'
  Say '     PARM('pgm_parms')'
  Say
  Say
  "SEND '"abnd pgmname "program error  rc =" retcode"     '"
  If userids <> '' Then
     "SEND '"Time() Date('E') ' ',
        abnd pgmname 'program error  rc =' retcode,
        "    ' USER("userids") LOGON"
  Return
```

MONITOR STARTED TASK PROCEDURE

Create a monitor started task procedure.

```
//*********************************************************************/
//* FUNCTION = MONITOR A DATAPROPAGATOR SUBSCRIPTION SET              */
//*********************************************************************/
//*
//*  START:  S MONITOR   OR   S MONITOR,PRM='.....'
//*  STOP :  C MONITOR
//*-------------------------------------------------------------------
//*  PARAMETERS: SSID     LOCAL DB2 SSID FOR DPROP
//*              QUAL     APPLY QUALIFIER
//*              SUBSET   APPLY SUBSCRIPTION SET NAME
//*              CAPT     CAPTURE TASK NAME
//*              APPLY    APPLY TASK NAME
//*              WAIT     WAIT TIME FOR EACH MONITOR CYCLE      (mins)
//*              MAXLAG   MAXIMUM SYNCH LAG TIME BEFORE WARNING (secs)
//*              MSGS     CONSOLE, TSO, BOTH or NONE
//*              NOTIFY   LIST OF TSO USERS TO RECEIVE WARNING MSGS
//*              END      TIME TO AUTOMATICALLY STOP           (hh:mm)
//*
//*  THE DEFAULT VALUES OF ALL PARAMETERS ARE SPECIFIED IN THE PARM
//*  ON THE EXEC STATEMENT. IF YOU WISH TO OVERRIDE ONE OR MORE OF THEM
//*  THEN USE THE 'PRM' PARAMETER ON THE START COMMAND, FOR EXAMPLE:
//*              S MONITOR,PRM='WAIT(10) END(19:00) NOTIFY(userid)'
//*-------------------------------------------------------------------
//MONITOR PROC PRM=''
//DPROPMON EXEC PGM=IKJEFT01,DYNAMNBR=20,
```

```
//    PARM=('%DPROPMON &PRM SSID(ssid) QUAL(qualifier) SUBSET(subset)',
//          'APPLY(apply-task) WAIT(5) MAXLAG(24Ø)')
//STEPLIB  DD DSN=ssid.RUNLIB.LOAD,DISP=SHR    <- DPRMON,WAITA in this
//         DD DSN=ssid.SDSNLOAD,DISP=SHR
//SYSEXEC  DD DSN=DPROPMON.LIB,DISP=SHR        <- DPROPMON in this
//DPRWORK  DD DSN=&&DPRWORK,DISP=(,DELETE),UNIT=SYSDA,
//            SPACE=(TRK,1),DCB=(RECFM=FB,LRECL=132,BLKSIZE=Ø)
//SYSTSPRT DD SYSOUT=T
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD DUMMY
```

CONCLUSION

Though simple, this monitor provides enough information to check that propagation is proceeding correctly and gives automatic warnings if it is not. You specify your goal of propagation lag as less than a particular number of seconds, then let the monitor warn you if that is exceeded.

This monitor was sent to the IBM DataPropagator Development Team. They said that it "looks very good", and I "don't need to change the SQL for Version 8 because IBM will be providing their own monitor". However, I believe that their monitor runs from Control Center on NT/ AIX etc, and not directly on MVS.

Ron Brown (Consultant)

How do I set up federated databases on DB2 UDB?

This article describes what federated database support is and how to set it up in a DB2/UDB environment.

What does federated database support give me? Well, in a nutshell, it lets me select from tables in two different databases in the same unit of work. These databases can be DB2 UDB on Unix and Windows, DB2 OS/390, DB2/400, or DB2/VM/VSE/DS (and Oracle, see below). If you are using DB2, then you need to be running UDB DB2 7.1 as a minimum. With a federated database set-up, you can only select from the participating databases. If you want to update tables, you need to use the passthru statement.

Let's look at an example to see how it works in practice. Suppose we want to select from table tab_aix (in database sample running on DB2 7.1 on an RS/6000) and table tab_dbwin1 in database dbwin1 running on Windows 2000 in an inner join select statement. This is a DB2/DB2 federated example.

The table tab_aix (on the RS/6000) has one row in it and is defined as follows:

```
>db2 connect to sample user xxxx using yyyyy
>db2 create table db2inst1.tab_aix (id int, name char(10))
>db2 insert into db2inst1.tab_aix values (1,'Mary')
```

To use federated database support you have to enable the dbwin1 database (this is because we will be issuing the select statement from this database). You cannot enable the database as such, you have to enable the instance in which the database resides. Issue the commands:

```
>db2 update dbm cfg using federated yes
>db2stop force
>db2start
```

(You need to stop/start the instance to pick up the update change.)

To set up a federated database on dbwin1, follow these steps:

- Create a wrapper.
- Create a server.
- Create a user mapping.
- Create a nickname.

To create a wrapper:

```
>db2 connect to dbwin1
>db2 create wrapper drda
```

The drda option for the wrapper name means all DB2 sources. If you wanted to connect to an Oracle database, you would use NET8. This is all documented in the *SQL Reference* manual under *Create wrapper*. (To connect to an Oracle database from DB2 you would need DB2 Connect Relational.)

To create a server that points to the DB2 system on the RS/6000:

```
>db2 create server db2aixs(1) type db2/6000(2) version 7.1(3) wrapper
drda(4) authorization xxxx(5) password yyyyyy(6) options (node
'TCP0000'(7), dbname 'sample'(8))
```

Ignore the numbers in brackets; they indicate:

1 The server name db2aixs can be any name you like (as long as one doesn't already exist in the dbwin1 database).

2 We want to select from the database on an RS/6000, hence the type.

3 This is the version of DB2 on the RS/6000.

4 This is the wrapper we created earlier.

5 This is the userid that we want to use on the RS/6000. This could be the DB2INST1 userid.

6 This is the password of the userid on the RS/6000.

7 This is the node name of the RS/6000. To find this value, do a >db2 list node directory.

8 This is the database on the RS/6000 that you want to select from.

This command takes a few seconds to execute, and it will hang if you:

- Enter db2/AIX (instead of db2/6000).
- Put the wrong version.

To create a user mapping for a valid DB2 user:

```
>db2 create user mapping for db2admin(1) server db2aixs(2) options
(remote_authid 'xxxx'(3), remote_password 'yyyyyy'(4))
```

Ignore the numbers in brackets; they indicate:

1 This is the sysadm userid on Windows (where dbwin1 is).

2 The server name (see above).

3 This is the userid on the RS/6000 (see above).

4 This is the password of the userid on the RS/6000 (see above).

To create a nickname for the table on the RS/6000:

```
>db2 create nickname db2admin.nick1(1) for
```

```
db2aixs(2).db2inst1(3).tab_aix(4)
```

Ignore the numbers in brackets; they indicate:

1 This is the schema and name of the nickname that you want to create.

2 This is the server name (see above).

3 This is the schema for the table on the RS/6000.

4 This is the table name on the RS/6000.

You have now finished setting up the federated database. Try to select from the table (remembering that we are still connected to dbwin1 with a userid of db2admin).

```
>db2 select * from nick1

ID          NAME
----------- ----------
          1 Mary

  1 record(s) selected.
```

What this shows is that you can select from one database (sample on the RS/6000) whilst connected to another (dbwin1 on Windows).

You can use the nickname in a join with a table (tab_dbwin1) on dbwin1, as shown below:

```
>db2 connect to dbwin1

>db2 create table tab_dbwin1 (id int, dept char(10))

>db2 insert into tab_dbwin1 values(1,'Ops')

>db2 select a.name, b.dept from nick1 a, tab_dbwin1 b where a.id = b.id

NAME       DEPT
---------- ----------
Mary     Ops

  1 record(s) selected.
```

Using the nickname, you can only select from the federated tables. If you try to insert using the nickname, you get:

```
>db2 insert into nick1 values(3,'Sue')
```

```
DB21034E The command was processed as an SQL statement because it was
not a valid Command Line Processor command. During SQL processing it
returned:
SQL30090N Operation invalid for application execution environment.
Reason code = “21”. SQLSTATE=25000
```

If you want to insert into a federated table you have to use the passthru statement, as shown below:

```
>db2 connect to dbwin1

>db2 set passthru db2aixs

>db2 insert into db2inst1.tab_aix values(2,'Fred')

>db2 set passthru reset

>db2 select * from nick1

ID          NAME
---------- ----------
          1 Mary
          2 Fred

  2 record(s) selected.
```

Note: you cannot passthru to two different databases at the same time – ie you can specify only one database in a passthru statement.

If you want to drop a wrapper, then the process is a reverse of the creation process, namely:

- Drop the nickname:

  ```
  >db2 drop nickname db2admin.nick1
  ```

- Drop the user mapping:

  ```
  >db2 drop user mapping for db2admin server db2aixs
  ```

- Drop the server:

  ```
  >db2 drop server db2aixs
  ```

- Drop the wrapper:

  ```
  >db2 drop wrapper drda
  ```

The catalog tables that contain information about wrappers are:

```
SYSCAT.SERVEROPTIONS
```

SYSCAT.USEROPTIONS
SYSCAT.WRAPOPTIONS
SYSCAT.WRAPPERS

CONNECTING TO OTHER DATA SOURCES

If you need to connect to other relational databases (such as NCR Teradata, Informix, etc) or other data sources such as IMS or VSAM files, you need a product such as Datajoiner. If you want to connect to Oracle, then you need DB2 Connect Relational.

I hope I have shown you how simple it is to set up a federated database system.

C Leonard
Freelance Consultant (UK)

Compare DDL for indexes

The following REXX code is similar in functionality to that in the article *Compare DDL for tables and columns* on page 37 of *DB2 Update*, Issue 117, July 2002.

This time the REXX program compares the indexes of the same tables in two different subsystems and reports discrepancies in the indexes such as index names, key columns, and orderings.

If a table exists in more than a couple of regions and if the DBA needs to monitor the indexes of the same tables in different regions, the program can be very useful to verify the synchronization of the indexes.

```
/****************************** REXX ***********************************/
/*   Compare the indexes between two subsystems                        */
/*                                                                     */
/*   SQLREX, an in-house developed REXX/SQL interface is used, but     */
/*   the program will still work without the interface if two input    */
/*   datasets are provided, which contain SQL output from              */
/*   each subsystem:                                                   */
/*                                                                     */
/*    SELECT A.IXNAME, A.COLNAME, A.COLSEQ, A.ORDERING                 */
/*      FROM SYSIBM.SYSKEYS A, SYSIBM.SYSINDEXES B                     */
```

```
/*      WHERE B.TBNAME IN ('table1'                                    */
/*                        ,'table2'                                    */
/*                        ,'table3'                                    */
/*                        ,..other tables..... )                       */
/*        AND B.CREATOR = index-creator                                */
/*        AND A.IXNAME = B.NAME                                        */
/*        AND A.IXCREATOR = B.CREATOR                                  */
/*      ORDER BY IXNAME, COLSEQ ;                                      */
/*                                                                     */
/*********************************************************************/
/*   Input parameters                                                  */
/*     subsys1 - DB2 subsystem 1                                       */
/*     subsys2 - DB2 subsystem 2                                       */
/*     indsn1  - dsn that contains the SQL output from subsys1.        */
/*               The contents of the dataset will be overwritten       */
/*               if REXX/SQL interface is used.                        */
/*     indsn2  - dsn that contains the SQL output from subsys2.        */
/*               The contents of the dataset will be overwritten       */
/*                 if REXX/SQL interface is used.                      */
/*     stmt    - Optional parameter.                                   */
/*               Will be needed when the REXX/SQL interface is used.   */
/*               It is the SQL statement as above.                     */
/*               The list of tables in the SQL SELECT will not be      */
/*                 needed if all the indexes of the creator are being*/
/*                 compared.                                           */
/*               If the two subsystems have exactly the same SQL       */
/*               then specify just one sql, otherwise it will need     */
/*               two SELECT statements delimited by a semicolon.       */
/*                                                                     */
/*********************************************************************/
Arg subsys1 indsn1 subsys2 indsn2 stmt
x = sysdsn(indsn1)
If x <> 'OK'
   Then call newalloc indsn1 indd1
   Else do
      Address tso
      "ALLOC DDNAME(indd1) DSNAME('"indsn1"') SHR REUSE " ,
         " RECFM(F B) LRECL(8Ø) BLKSIZE(312Ø)"
   End
y = sysdsn(indsn2)
If y <> 'OK'
   Then call newalloc indsn2 indd2
   Else do
      Address tso
      "ALLOC DDNAME(indd2) DSNAME('"indsn2"') SHR REUSE " ,
         " RECFM(F B) LRECL(8Ø) BLKSIZE(312Ø)"
   End

If stmt <> '' then do
   Parse var stmt stmt1 ';' stmt2
```

```
   Parse var stmt1 stmt11 '<' strconst '>' stmt12
   Do while  strconst <> ''
      stmt1 = stmt11 "'" || strconst || "'" stmt12
      Parse var stmt1 stmt11 '<' strconst '>' stmt12
   End
   If stmt2 <> '' then do
      Parse var stmt2 stmt21 '<' strconst '>' stmt22
      Do while  strconst <> ''
         stmt2 = stmt21 "'" || strconst || "'" stmt22
         Parse var stmt2 stmt21 '<' strconst '>' stmt22
      End
   End
   Else stmt2 = stmt1
End
Else do
   say "No SQL provided. INDSN1 and INDSN2 will be used as it is.."
   signal readin
End

"SQLREX CONNECT "subsys1
If SQL_CC <> Ø then do
   rc = SQL_ERROR()
   say "*** Connection error to" subsys1
   say "*** INDSN1 file will be used as it is ...."
End
Else do
   "SQLREX DECLARE D1 CURSOR FOR "stmt1
   If SQL_CC <> Ø then do
      rc = SQL_ERROR()
      say "*** SQL execution error in" subsys1
      say "*** INDSN1 file will be used as it is..."
   End
   Else do
      rec1. = ''
      do i = 1 to D1
         rec1.i = D1_IXNAME.i D1_COLNAME.i D1_COLSEQ.i D1_ORDERING.i
      End
      Address TSO
      "EXECIO * DISKW indd1 (stem rec1. finis"
   End
End

"SQLREX CONNECT "subsys2
If SQL_CC <> Ø then do
   rc = SQL_ERROR()
   say "*** Connection error to" subsys2
   say "*** INDSN2 file will be used as it is ...."
End
Else do
   "SQLREX DECLARE D2 CURSOR FOR "stmt2
   If SQL_CC <> Ø then do
```

```
      rc = SQL_ERROR()
      say "*** SQL execution error in" subsys2
      say "*** INDSN2 file will be used as it is..."
   End
   Else do
      rec2. = ''
      do i = 1 to D2
         rec2.i = D2_IXNAME.i D2_COLNAME.i D2_COLSEQ.i D2_ORDERING.i
      End
      Address TSO
      "EXECIO * DISKW indd2 (stem rec2. finis"
   End
End

readin:
Number_of_lines1 = Ø
Number_of_lines2 = Ø
Number_of_indexes1 = Ø
Number_of_indexes2 = Ø
Number_of_keys1. = Ø
Number_of_kyes2. = Ø
ixnames1. = ''
ixnames2. = ''
keys1. = ''
keys2. = ''
ordering1. = ''
ordering2. = ''

/*************************************************/
/*               Read in the input               */
/*************************************************/
"EXECIO * DISKR indd1 (FINIS"
Number_of_lines1 = queued()
i = Ø
Do Number_of_lines1
   Pull w1 w2 w3 w4
   If w3 = 1 then do
      i = i + 1
      Number_of_keys1.i = w3
      ixnames1.i = w1    /* i-th index in subsys 1 */
      keys1.i.1 = w2     /* 1st key of i-th index in sybsys 1 */
      ordering1.i.1 = w4
      end
   else if w3 <> '' then do
      Number_of_keys1.i = w3
      keys1.i.w3 = w2    /* w3-th key of i-th index in sybsys 1 */
      ordering1.i.w3 = w4
      end
      else nop
End
Number_of_indexes1 = i
```

```
say ' '
say '*****************************************************'
say '**           List of indexes compared                *'

"EXECIO * DISKR indd2 (FINIS"
Number_of_lines2 = queued()
j = Ø
Do Number_of_lines2
   Pull w1 w2 w3 w4
   If w3 = 1 then do
      j = j + 1
      Number_of_keys2.j = w3
      ixnames2.j = w1
      keys2.j.1 = w2
      ordering2.j.1 = w4
   End
   Else if w3 <> '' then do
      Number_of_keys2.j = w3
      keys2.j.w3 = w2
      ordering2.j.w3 = w4
   End
   Else nop
End
Number_of_indexes2 = j

say left('*  Number of indexes : ' ||subsys1||' '||i|| ,
         ' '||subsys2||' '||j,52) '*'
say '*****************************************************'
say ' '
say left(subsys1,25) subsys2
say left('-------------',25) '-------------'

i = 1
j = 1
Do until ((i > Number_of_indexes1) | (j > Number_of_indexes2))
   If ixnames1.i < ixnames2.j then do
       say left(ixnames1.i,25) '***'
       i = i + 1
   End
   Else if ixnames1.i > ixnames2.j then do
       say left('***',25) ixnames2.j
       j = j + 1
       End
       Else do
          say left(ixnames1.i,25) ixnames2.j
          i = i + 1
          j = j + 1
       End
End
If i > Number_of_indexes1 then
   Do while (j <= Number_of_indexes2)
```

```
      say left('***',25) ixnames2.j
      j = j + 1
   End
   Else if j > Number_of_indexes2 then
      Do while (i <= Number_of_indexes1)
         say left(ixnames1.i,25) '***'
         i = i + 1
      End
      Else nop
say ' '
say ' '

/*****************************************************/
/*    Compare the keys of the indexes                */
/*****************************************************/
compare_index:
i = 1
j = 1
dx1 = 0
dx2 = 0
diffxn1. = ''
diffxn2. = ''
ds = 0
diffxs. = ''
dk = 0
diffxk. = ''
dr = 0
diffxr. = ''

Do while (( i <= Number_of_indexes1) & (j <= Number_of_indexes2))
   If ixnames1.i < ixnames2.j then do
      dx2 = dx2 + 1
      diffxn2.dx2 = left(ixnames1.i,18) ' is missing in ' subsys2
      i = i + 1
      end
      else if ixnames1.i > ixnames2.j then do
         dx1 = dx1 + 1
         diffxn1.dx1 = left(ixnames2.j,18) ' is missing in ' subsys1
         j = j + 1
         end
         else do  /* same index names */
            call compare_key
            i = i + 1
            j = j + 1
            end
End

Select
when (i > Number_of_indexes1 & j <= Number_of_indexes) then
   do until (j > Number_of_indexes2)
      dx1 = dx1 + 1
```

```
      diffxn1.dx1 = left(ixnames2.j,18) ' is missing in ' subsys1
      j = j + 1
   end
when (i <= Number_of_indexes1 & j > Number_of_indexes) then
   do until (i > Number_of_indexes1)
      dx2 = dx2 + 1
      diffxn2.dx2 = left(ixnames1.i,18) ' is missing in ' subsys2
      i = i + 1
   end
otherwise nop
End

/*********************************************************/
/**      Write out the report                           **/
/*********************************************************/
say '***********************************************'
say '**               Missing indexes             **'
say left('**       '||subsys1||'  '||dx1||'  ',
         subsys2||'  '||dx2,44) '**'
say '***********************************************'
If dx1 > Ø then
   Do i = 1 to dx1
      say left(' '||i||'.',5) diffxn1.i
   End
Else nop
say ' '
If dx2 > Ø then
   Do i = 1 to dx2
      say left(' '||i||'.',5) diffxn2.i
   End
Else nop
say ' '
say '*********************************************************'
say '**       Indexes having different number of keys      ***'
say '*********************************************************'
If ds = Ø then do
   say ' '
   say 'None'
End
Else do i = 1 to ds
   say left(' '||i||'.',5) diffxs.i
End
say ' '
say '*****************************************************************'
say '**          Same number of keys but different keys              *'
say '*****************************************************************'
If dk = Ø then do
   say ' '
   say 'None'
End
Else do i = 1 to dk
```

```
   say left(' '||i||'.',5) diffxk.i
End
say ' '
say '************************************************************'
say '**        Different ordering "A" or "D"                   **'
say '************************************************************'
If dr = Ø then do
   say ' '
   say 'None'
   say ' '
   say ' '
End
Else do i = 1 to dr
   say left(' '||i||'.',5) diffxr.i
End

Address tso
"FREE FI(indd1)"
"FREE FI(indd2)"
exit (Ø)

compare_key:
/*********************************************************************/
/*   Compare the keys of the same indexes from the two subsystems  */
/*       keys1.i.k  :  k-th key of the i-th index in subsys 1      */
/*       keys2.j.k  :  k-th key of the j-th index in subsys 2      */
/*********************************************************************/
If Number_of_keys1.i <> Number_of_keys2.j
   then do
      ds = ds + 1
      diffxs.ds = left(ixnames1.i,18) ' has ',
                                    Number_of_keys1.i ' in' subsys1 ,
                                ' ' Number_of_keys2.j ' in' subsys2
   End
   Else nop
k = 1
l = 1
Do until ((k > Number_of_keys1.i) | (l > Number_of_keys2.j))
   If keys1.i.k = keys2.j.l then do
      call compare_ordering
      k = k + 1
      l = l + 1
   End
   Else do
      dk = dk + 1
      diffxk.dk = left(ixnames1.i,18) k left(keys1.i.k,18),
                                          ' in ' subsys1 not matching
      leave
   End
End
return
```

```
compare_ordering :
If ordering1.i.k <> ordering2.j.l
   then do
      dr = dr + 1
      diffxr.dr = left(ixnames1.i,18) ' in' subsys1,
                ' has unmatching ordering for key k keys1.i.k'
   End
   else nop
return

newalloc: procedure
   Arg dsn ddn
   Address TSO
   "ALLOC DDNAME("ddn") DSNAME('"dsn"') NEW CATALOG SPACE(5,5) TRACKS",
      " UNIT(DISK) RECFM(F B) LRECL(132) BLKSIZE(13200)"
return

SQL_ERROR:
    SAY "SQL_DB2SSN = '"SQL_DB2SSN"'"
    SAY "SQL_CC = '"SQL_CC"'"
    SAY "SQL_RC = '"SQL_RC"'"
    SAY "SQL_REASON = '"SQL_REASON"'"
    SAY "SQL_MESSAGE = '"SQL_MESSAGE"'"

RETURN 0
```

HOW TO RUN THE PROGRAM

The following IKJEFT01 will run the REXX program as a batch job.

It compares the indexes for all the tables in the two subsystems, DB2B and DB2C, for the index creator ‘theixcreatr’.

Since the SQL SELECT statement is provided in the JCL, the two input datasets, myuid.SYSKEYS..., will be overwritten by the output of the SQL SELECT and the original contents cannot be used.

The program is located in a PDS, userid.DB2GEN, in a member CMPIX.

Note that the index creator has brackets <>.

```
//N1234SP1   JOB (1234SP),'SAM PARK,1234',CLASS=A,
//  NOTIFY=N1234SP,MSGCLASS=X,MSGLEVEL=(1,1),REGION=6M
//ST1  EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
```

```
//SYSTSIN DD *
   EXEC DB2GEN(CMPIX) 'DB2C myuid.SYSKEY.DB2C.DØ21Ø1 DB2B
myuid.SYSKEY.DB2B.DØ21Ø1 SELECT A.IXNAME, A.COLNAME, A.COLSEQ,
A.ORDERING FROM SYSIBM.SYSKEYS A, SYSIBM.SYSINDEXES B WHERE B.CREATOR =
<theixcreatr> AND A.IXNAME = B.NAME AND A.IXCREATOR = B.CREATOR ORDER BY
IXNAME, COLSEQ ; '
/*
```

OUTPUT FROM THE PROGRAM

The output has been edited to reduce the length for publication. The first part, list of indexes compared, will display all the existing indexes for the tables in the two subsystems and will show any missing indexes by '***'.

```
1ACFØC038 ACF2 LOGONID ATTRIBUTES HAVE REPLACED DEFAULT USER ATTRIBUTES
 READY
   EXEC DB2GEN(CMPIX) 'DB2C myuid.SYSKEY.DB2C.DØ21Ø1 DB2B
myuid.SYSKEY.DB2B.DØ21Ø1 SELECT A.IXNAME, A.COLNAME, A.COLSEQ,
A.ORDERING FROM SYSIBM.SYSKEYS A, SYSIBM.SYSINDEXES B WHERE B.CREATOR =
<theixcreatr> AND A.IXNAME = B.NAME AND A.IXCREATOR = B.CREATOR ORDER BY
IXNAME, COLSEQ ; '

 ******************************************************
 **           List of indexes compared               *
 *  Number of indexes : DB2C 775 DB2B 83Ø             *
 ******************************************************

 DB2C                    DB2B
 --------------          ---------------
 ACCTKINDØX              ***
 ACCTØX                  ACCTØX
 ACCT1X                  ACCT1X
 ***                     ACMPTYØX
 GLDALC1X                GLDALC1X
 GLDALC2X                GLDALC2X
 GLNACTØX                GLNACTØX
 GLNACT1X                GLNACT1X
 ***                     GLNTØX
 ***                     GLOONTØX
 ***                     GLPRODØX
 GMEDIACDØX              ***
 ***                     GNTRCLØX
 GRPADRØX                GRPADRØX
 GRPADR1X                GRPADR1X
 GRPADR2X                GRPADR2X


 ***********************************************
```

```
**               Missing indexes             **
**       DB2C  287   DB2B  232                 **
************************************************
 1.   ACMPTYØX            is missing in  DB2C

 1.   ACCTKINDØX          is missing in  DB2B

*********************************************************
**       Indexes having different number of keys      ***
*********************************************************
 1.   ACTRCKØX            has  3  in DB2C   4  in DB2B
 2.   ACTRCK1X            has  7  in DB2C   6  in DB2B
 3.   ACTRCK2X            has  4  in DB2C   3  in DB2B
 4.   ADDRØX              has  2  in DB2C   3  in DB2B
 5.   ADRRELØX            has  4  in DB2C   5  in DB2B

*****************************************************************
**          Same number of keys but different keys              *
*****************************************************************
 1.   ACTRCKØX           1 WH_HCA_ID           in  DB2C NOT MATCHING
 2.   ACTRCK1X           3 ACCT_NUM            in  DB2C NOT MATCHING
 3.   ACTRCK2X           1 WH_PARTTN_NUM       in  DB2C NOT MATCHING
 4.   ADDRØX             1 WH_ADDR_ID          in  DB2C NOT MATCHING
 5.   ADRRELØX           1 WH_ADDR_1_ID        in  DB2C NOT MATCHING

*********************************************************
**        Different ordering "A" or "D"                **
1*********************************************************

 None


 READY
 END
```

Samuel Park
DBA Consultant
Ricoh (USA)

Code from individual articles of *DB2 Update*, and complete issues in Acrobat PDF format, can be accessed on our Web site, at:

http://www.xephon.com/db2

You will be asked to enter a word from the printed issue.

IBM has announced Version 8.1 of its DB2 Everyplace to help provide mobile workers with access to enterprise applications and data.

The small-footprint relational database and synchronization server enables enterprise applications and data to be extended to mobile devices. It can also be embedded into devices and appliances.

It's capable of high-performance data queries handling large volumes of mobile data and users via a high-powered database engine in a server in a connected or disconnected mode. It also delivers information encrypted on the handheld device.

The associated Sync Server exchanges data between enterprise data sources and mobile devices and also allows real-time queries to be performed. There's also multiple server support and automatic upgrade deployment. V8.1 enables application development using industry-standard tools and open APIs. Applications can be written in Java or C/C++ code using WebSphere Studio Device Developer or Metrowerks CodeWarrior. The DB2 Everyplace SDK supports Java, C/C++, and Visual Basic. The Mobile Application Builder is included for rapid application development without, it's claimed, writing a single line of code.

IBM's AppForge plug-in can be used to create mobile applications using Visual Basic.

The SDK is included as part of DB2 Universal Developer's Edition and is also available free for download from the DB2 Everyplace Web site.

V8.1 comes in two editions: DB2 Everyplace Database Edition, which includes the relational database for multiple platforms, and the Enterprise Edition, which also includes the Sync Server.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/data/db2.

* * *

IBM has announced System Automation for OS/390 V2.2 (SA OS/390), which provides policy-based self-healing. SA OS/390 also has an easy-to-use GUI, and 'canned' automation for DB2, IMS, CICS, and TWS.

IBM has also announced Version 3.2 of its Enterprise COBOL for z/OS and OS/390, now with support for new advanced functions in Debug Tool, plus enhanced Java interoperability and improved COBOL DB2 Unicode support.

The software is geared to making use of existing COBOL resources and integrating them with new Web-oriented business processes.

The DB2-COBOL co-processor support has been enhanced so that when host variables are specified in SQL statements, it's no longer necessary in most cases to specify the code pages for the host variables declared with USAGE NATIONAL, DISPLAY, or DISPLAY-1 using the SQL DECLARE VARIABLE statements.

For further information contact your local IBM representative.
URL: http://www.ibm.com/zseries/software/sa/
URL: http://www.ibm.com/software/ad/cobol