63

DB2

January 1998

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: xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about DB2, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all DB2 users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *DB2 Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

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.

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £235.00 in the UK; $350.00 in the USA and Canada; £241.00 in Europe; £247.00 in Australasia and Japan; and £245.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £20.00 ($30.00) each including postage.

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

DB2 Image Copy analyser

As a DB2 DBA, some of my responsibilities include the following:

- Identifying the current space used by tablespaces in terms of pages. This helps in anticipating any future growth of data and also in estimating current DASD storage.
- Analysing the free space available in existing tablespaces. This helps in estimating storage needs of future inserts. It also tells us if we need to change our storage parameters like PCTFREE and FREEPAGE for the Tablespace before next REORG takes place.
- Finding the percentage of tablespace pages changed since the last Image Copy was taken. This tells us how much data has changed, in terms of number of pages since last back-up, which helps us in determining our back-up strategy – ie how frequently to schedule our back-ups and also which tablespaces qualify for taking incremental back-ups. For example, take incremental back-ups of only those tablespaces for which the percentage change is less than 10.

The utility, DB2 Image Copy Analyser, is a REXX program that takes the input from DB2 IMAGECOPY jobs. When we run IMAGECOPY for our tablespaces, the SYSPRINT dataset for the job gives very useful information. This utility reads the dataset associated with DD SYSPRINT and produces a report, as shown in Figure 1.

Tablespace name	Partition number	Number of pages	Percentage change	Avg percentage of Free Space	Elapsed time
DB001.TS0001	1	40	12.11	23.01	00:00:25
DB001.TS0001	2	60	6.01	12.45	00:00:34
DB002.TS0002		80	20.89	1.01	00:00:45

Figure 1: Image Copy report

IMAGE COPY ANALYSER

```
/*REXX*/
/*  Initialize indicators    */
rec1=' '
rec2=' '
first_ind=Ø
first_rec=Ø
n_pages=''
ts_name=''
part_num=''
avg_free_space=''
perc_change=''
ti_me=''
i=Ø
j=Ø
k=Ø
file_num=Ø
fileout =  'YOUR.IMAGECOPY.REPORT'
if sysdsn("'"fileout"'") ¬= "OK" then
do
  say 'Output file ' fileout ' does not exist.'
  say
  exit
end

/* Allocate Output Report dataset   */
"ALLOC DA('"||fileout||"')  F(DATAOUT) OLD REUSE"
"EXECIO Ø DISKW DATAOUT (OPEN"
call main_rtn
"EXECIO Ø DISKW DATAOUT (FINIS"
"FREE F(DATAOUT)"
exit
main_rtn:
file_char=''
eof='NO'
rec1=' '
rec2=' '
first_ind=Ø
n_pages=''
ts_name=''
perc_change=''
ti_me=''
i=Ø
j=Ø
k=Ø
eof='NO'
INLIST.   = ""
INLIST.Ø  = Ø
VAR.   = ""
VAR.Ø  = Ø
```

```
OUTLIST.   = ""
OUTLIST.Ø  = Ø
 filename=''
 filename='YOUR.SYSPRINT.DATASET'
if sysdsn("'"filename"'") ¬= "OK" then
do
  say 'Input file ' filename ' does not exist.'
  say
  return
end
"ALLOC DA('"||filename||"') F(DATAIN)  SHR REUSE"
"EXECIO Ø DISKR DATAIN  (OPEN"
do while eof='NO'
    "EXECIO 1 DISKR DATAIN (STEM INLIST."
    if RC=2 then
        eof='YES'
    else
        call  process_rtn
end
call process_print
"EXECIO Ø DISKR DATAIN  (FINIS"
"FREE F(DATAIN)"
return
process_rtn:
    a=''
    b=''
    c=''
    d=''
    e=''
    f=''
    rec1=inlist.1
    if (substr(rec1,1,1Ø) ¬= 'ØDSNUØ5ØI ' ) & ( first_ind = Ø ) then
        return
    if substr(rec1,1,23) = 'ØDSNUØ5ØI   DSNUGUTC - '  then
        do
            if first_ind=Ø then
                do
                    first_ind=1
                    if first_rec=Ø then
                    do
                     first_rec=1
                     rec2=''
                     rec2='Table Space     Partition     Number'
                     rec2=rec2||'   Percentage'||'    '
                     rec2=rec2||'Avg Percantage    Elapsed '
                     rec2=rec2||'            '||date()
                     call process_write
                     rec2=''
                     rec2='Name             Number       of Pages'
                     rec2=rec2||'   Change '||'   '
                     rec2=rec2||'  of Free Space     time'
```

```
                    call process_write
                    rec2=''
                    rec2='_____________________________________________'
                    rec2=rec2||'_____________________________________________'
                    call process_write
                    rec2=' '
                    call process_write
                    call process_write
                   end
               end
           else
               call process_print
           if pos('COPY TABLESPACE ',rec1) ¬=Ø then
               do
                   parse value rec1 with a b c d e ts_name f
                   if subword(f,1,1)='DSNUM' then
                       part_num=subword(f,2,1)
                   else
                       part_num='   '
               end
       end
rec1=strip(rec1)
n=Ø
n=pos('NUMBER OF PAGES=',rec1)
if n ¬= Ø then
   do
       n=n+16
       rec1=substr(rec1,n)
       n_pages=strip(rec1)
       return
   end
n=Ø
n=pos('AVERAGE PERCENT FREE SPACE PER PAGE = ',rec1)
if n ¬= Ø then
   do
       n=n+37
       rec1=substr(rec1,n)
       avg_free_space=strip(rec1)
       return
   end
n=Ø
n=pos('PERCENT OF CHANGED PAGES = ',rec1)
if n ¬= Ø then
   do
       n=n+27
       rec1=substr(rec1,n)
       perc_change=strip(rec1)
       return
   end
n=Ø
```

```
  n=pos('ELAPSED TIME=',rec1)
  if n ¬= 0 then
     do
         n=n+13
         rec1=substr(rec1,n)
         ti_me=strip(rec1)
         return
     end
return 0
process_print:
    rec2= ' '
    rec2 = substr(ts_name,1,17)||'  '||substr(part_num,1,3)
    rec2 = rec2||'        '||substr(n_pages,1,8)
    rec2 = rec2||'    '||substr(perc_change,1,5)
    rec2 = rec2||'          '||substr(avg_free_space,1,5)
    rec2 = rec2||'          '||substr(ti_me,1,8)
    call process_write
    n_pages=''
    ts_name=''
    part_num=''
    avg_free_space=''
    perc_change=''
    ti_me=''
return 0
process_write:
    outlist.1 = rec2
    "EXECIO 1 DISKW DATAOUT (STEM OUTLIST."
return 0
```

Sharad Kumar Pande
DBA (USA)

Call for papers

Why not share your expertise and earn money at the same time? *DB2 Update* is looking for REXX EXECs, macros, program code, etc, that experienced DB2 users have written to make their life easier. We will publish them (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

Automated JCL generation for DB2 utilities – part 4

This month we conclude the code that automatically generates JCL to test DB2 utilities.

SLQSPDB ASSEMBER

```
//<YOUR JOBCARDS>
//*************************************************************
//*   OUR DB2 EXIT LIB:  DB2T.PBDA.SDSNEXIT                   *
//*   OUR DB2 LOAD LIB:  DB2T.PBDA.SDSNLOAD                   *
//*   OUR DB2 RUN  LIB:  DB2T.PB.RUNLIB.LOAD                  *
//*   OUR DBRM     LIB:  DA.PB.DBRMLIB                        *
//*   OUR DB2         :  DB2T                                 *
//*************************************************************
//JOBLIB  DD  DISP=SHR,
//            DSN=DB2T.PB.SDSNEXIT
//        DD  DISP=SHR,
//            DSN=DB2T.PB.SDSNLOAD
//PC      EXEC PGM=DSNHPC,PARM='HOST(ASM),SOURCE',REGION=4Ø96K
//DBRMLIB DD DSN=DA.PB.DBRMLIB(SQLSPDB),
//         DISP=SHR
//STEPLIB  DD  DISP=SHR,DSN=DB2T.PB.SDSNEXIT
//         DD  DISP=SHR,DSN=DB2T.PB.SDSNLOAD
//SYSCIN   DD  DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(8ØØ,(1Ø,1Ø))
//SYSLIB DD    DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(8ØØ,(1Ø,1Ø),,,ROUND),UNIT=SYSDA
//SYSIN    DD  *
SQLSPDB  TITLE ' REXX INTERFACE WITH DB2 - CAF'
SQLSPDB  CSECT
*********************************************************************
* FUNCTION : THE RESULTS FROM THE SPECIFIED ESQL QUERY ARE RETURNED  *
*            AS REXX VARIABLES.                                     *
*            THE VARIABLE NAMES ARE THOSE ATTRIBUTES RESULTING      *
*            FROM THE ESQL QUERY.                                   *
*            IF A NON-SELECT SQL STATEMENT IS ENTERED NO OUTPUT IS  *
*            RETURNED, BUT THE COMMAND IS EXECUTED.                 *
*            A MAXIMUM OF 9 FUNCTIONS (LIKE -SUM- OR -COUNT-) ARE   *
*            ALLOWED.                                               *
*            IN CASE OF A NON-SELECT SQL COMMAND, YOU CAN HAVE ONE  *
*            (1) HOSTVARIABLE IN THE SQL STATEMENT (REPRESENTED BY A *
```

```
*            ?). PUT THE VALUE OF THE HOSTVARIABLE IN REXX VARIABLE   *
*            HOSTVAL.                                                 *
*            A HOSTVARIABLE IN A SELECT SQL STATEMENT WILL RESULT IN  *
*            SQLCODE -313.                                            *
*            THIS PROGRAM USES THE DB2 CAF INTERFACE AND SO CAN       *
*            EXECUTE OUTSIDE THE DB2 ENVIRONMENT (HOWEVER, SQL MUST   *
*            BE AVAILABLE).                                           *
*            THIS PROGRAM IS WRITTEN FOR THE SP/DB DB2 AIDS ONLY.     *
*            NO GUARANTEES ARE MADE!!!!!!!!!                          *
*            THERE IS A KNOWN BUG IN THIS PROGRAM! DO NOT ATTEMPT     *
*            TO PERFORM ARITHMETIC FUNCTIONS LIKE SUM(COLUMN)/1Ø IN   *
*            THIS PROGRAM. IT WILL MESS UP THE PRECISION!             *
*            TRY AND FIX IT. IF YOU DO, MAIL IT TO ME!                *
* PLANNAME : SQLSPDB                                                  *
* INPUT VARIABLE   : <DB2V>     : DB2 SYSTEM (DEFAULT IS DB2T)        *
*                  : <SQLQUERY> : SPECIFIC SQL QUERY                  *
*                  : <HOSTVAL>  : OPTIONAL HOST VARIABLE              *
* OUTPUT VARIABLES : COLUMNNAME(J).I (I = ROW)                        *
*                    <_VN.>.J   : COLUMN NAMES                        *
*                    <_VN.Ø>    : # OF COLUMNS                        *
*                    <_NROWS>   : # OF SELECTED ROWS                  *
*                    <_VN1> THRU <_VN9> FOR EVERY FUNCTION THATS USED *
*                    <_REASON>  : REASONCODE (ONLY IF BAD CONNECT)    *
* RETURNCODES      : Ø    -   OK                                      *
*                    <N>  -   REXX OR SQL CODE                        *
*                    99   -   BAD CONNECT TO DB2                      *
**********************************************************************
*        REGISTERS:                                                   *
*        R2     WORK                                                  *
*        R3     WORK                                                  *
*        R4     WORK                                                  *
*        R5     WORK                                                  *
*        R6     *<SQL> # OF COLUMNS                                   *
*        R7     A(SQLVARN)                                            *
*        R8     A(SQLDA)                                              *
*        R9     A(SQLCA)                                              *
*        R1Ø    TYPE POINTER                                          *
*        R11    FREE                                                  *
*        R12    BASE                                                  *
*        CO-AUTHOR: VALENTINO - AUV SP/DB @                           *
**********************************************************************
         REGEQ
         BEGIN SAVE=SA,BASE=(R11,R12)
         B     SA_END
SA       DS    18A
SA_END   DS    ØH
         LOAD  EP=IRXEXCOM               LOAD IRXEXCOM
         ST    RØ,AIRXEXCOM              SAVE ENTRYPOINT
* OBTAIN DB2 SYSTEM
```

```
* INITIALIZE IT TO TEST DB2 (DB2T)
* IF NO DB2V PARM, TAKE IT AS DEFAULT
         MVC   SSID,=C'DB2T'
         LA    R5,IRX_SHVBLOCK
         USING SHVBLOCK,R5
         MVI   SHVCODE,SHVFETCH
         MVC   SHVBUFL,=A(L'DB2V)
         MVC   SHVVALA,=A(DB2V)
         MVC   VN,=C'DB2V '
         BAL   R14,GETVNL            GET LENGTH OF <VN>
         ST    RØ,SHVNAML            L(<VN>)
         MVC   SHVNAMA,=A(VN)        A(<VN>)
         L     R15,AIRXEXCOM
         CALL  (15),(IRX_IRXEXCOM,Ø,Ø,IRX_SHVBLOCK),VL
         LTR   R15,R15               REXX RETURN CODE
         BNZ   NODBPRM               PARAMETER MISSING, TAKE DEFAULT
         L     R1,SHVVALL            L(PARAMETER)
         STH   R1,DB2VAR
         MVC   SSID(4),DB2V
NODBPRM  EQU   *
* OBTAIN <SQLQUERY>
         LA    R5,IRX_SHVBLOCK
         USING SHVBLOCK,R5
         MVI   SHVCODE,SHVFETCH
         MVC   SHVBUFL,=A(L'SQLQUERY)
         MVC   SHVVALA,=A(SQLQUERY)
         MVC   VN,=C'SQLQUERY '
         BAL   R14,GETVNL            GET LENGTH OF <VN>
         ST    RØ,SHVNAML            L(<VN>)
         MVC   SHVNAMA,=A(VN)        A(<VN>)
         L     R15,AIRXEXCOM
         CALL  (15),(IRX_IRXEXCOM,Ø,Ø,IRX_SHVBLOCK),VL
         LTR   R15,R15               REXX RETURN CODE
         BNZ   BLOWREXX              PARAMETER ERROR
         L     R1,SHVVALL            L(PARAMETER)
         STH   R1,SELECT
*        CONNECT TO DB2
         CALL  DSNALI,(CONNECT,SSID,TECB,SECB,RIBPTR),VL,MF=(E,CAFCALL)
         ST    R15,RETCODE
         ST    RØ,REASCODE
         CLC   RETCODE,=F'Ø'
         BNE   BLOW                  DB2 CONNECT ERROR
         CALL  DSNALI,(OPEN,SSID,PLAN),VL,MF=(E,CAFCALL)
         ST    RØ,REASCODE
         ST    R15,RETCODE
         CLC   RETCODE,=F'Ø'
         BNE   BLOWCON               DB2 PLAN ERROR
* OBTAIN POSSIBLE HOST VARIABLE <HOSTVAR>
         LA    R5,IRX_SHVBLOCK
```

```
         USING SHVBLOCK,R5
         MVI   SHVCODE,SHVFETCH
         MVC   SHVBUFL,=A(L'HOSTVAL)
         MVC   SHVVALA,=A(HOSTVAL)
         MVC   VN,=C'HOSTVAL '
         BAL   R14,GETVNL            GET LENGTH OF <VN>
         ST    RØ,SHVNAML            L(<VN>)
         MVC   SHVNAMA,=A(VN)        A(<VN>)
         L     R15,AIRXEXCOM
         CALL  (15),(IRX_IRXEXCOM,Ø,Ø,IRX_SHVBLOCK),VL
         LTR   R15,R15               TEST REXX RETURN CODE
         BNZ   DB2ST                 <> Ø, NO HOSTVAR SUPPLIED
         MVC   HOSTSW,YES            YES, THERE IS
         L     R1,SHVVALL            L(PARAMETER)
         STH   R1,HOSTVAR            STORE LENGTH
* DETERMINE NO OF COLUMNS IN TABLE
DB2ST    DS    ØH
         LA    R9,SQL_CA
         USING SQLDSECT,R9
         LA    R8,SQL_DA
         USING SQLDA,R8
* DUMMY PREPARE
         EXEC  SQL PREPARE S1 FROM :SELECT
         BAL   R14,CHECK_SQL
         MVC   SQLN,=H'Ø'            RETURN COUNT ONLY
* DESCRIBE
         EXEC  SQL DESCRIBE S1 INTO :SQL_DA
         BAL   R14,CHECK_SQL
* ANALYSE DESCRIPTOR AND ACQUIRE STORAGE
* <SQLD>: NO. OF COLUMNS
         LH    R2,SQLD
         MVC   RC,=F'Ø'              RESET RETURN CODE
         LTR   R2,R2
         BZ    A5ØØ                   :NON-SELECT
         LH    R3,SQLD
         MH    R2,=AL2(SQLVARN_SIZE)
         LA    R2,(SQLVAR-SQLDA)(R2)           +L(FIXED HDR)
* R2: TOTAL COLUMN SIZE
         ST    R2,COLSIZE
         GETMAIN EU,LV=(2),A=A_SQLDA
         L     R8,A_SQLDA
* PERFORM DESCRIBE WITH CORRECT LENGTH
         STH   R3,SQLD
         STH   R3,SQLN
         ST    R2,SQLDABC
* DESCRIBE
         EXEC  SQL DESCRIBE S1 INTO :SQLDA
         BAL   R14,CHECK_SQL
* BUILD ROW BUFFER
```

```
         LH    R6,SQLD             NO OF OCCURRENCES (COLUMNS)
         LTR   R6,R6
         BZ    EOP                 NO ENTRIES
         LA    R7,SQLVAR
         USING SQLVARN,R7
         SR    R4,R4               ZEROIZE BUFFER SIZE ACCUMULATOR
* OUTPUT NO OF COLUMNS (<_VN.Ø>)
         CNOP  Ø,4
         CVD   R6,D
         MVC   WK,=X'FØ202020202020Ø2Ø'
         ED    WK,D+4
         LA    R3,WK                  A(DATA)
         MVC   VL,=A(L'WK)           L(DATA)
         MVC   VNINDEX,=C'.Ø'
         MVC   VLINDEX,=A(2)
         MVC   VN,=C'_VN '
* SET DATA INTO RESS VARIABLE
         BAL   R14,SETVAR
A23Ø     DS    ØH
* COLUMN TYPE
         BAL   R14,GET_TYPE
* R2: DATA WIDTH
* R3: FIELD SIZE
* R1Ø: DSECT_TYPE ENTRY
         USING DSECT_TYPE,R1Ø
         LA    R4,2(R3,R4)          ACCUMULATE TOTAL BUFFER SIZE
         AP    INDEX,=P'1'          INCREMENT INDEX
* CONVERT INDEX TO FORM: .N
         MVC   VNINDEX,=X'4Ø2Ø2Ø2Ø2Ø2Ø212Ø'
         LA    R1,VNINDEX+7
         EDMK  VNINDEX,INDEX
         BCTR  R1,Ø
         MVI   Ø(R1),C'.'
         MVC   VNINDEX,Ø(R1)
         LA    RØ,VNINDEX+L'VNINDEX
         SR    RØ,R1
         ST    RØ,VLINDEX           L(INDEX)
         MVC   VN,=C'_VN '          NAME PREFIX
         LA    R3,SQLNAME+2         COLUMNS NAME
         LH    RØ,SQLNAME
* NEW CODE FOR UNNAMED COLUMNS (EG COUNT(*) )
* TEST WHETHER  NULL COLUMN NAME
         LH    RØ,SQLNAME           LOAD COLUMN NAME
         LTR   RØ,RØ
         BNZ   A231                 NO COLUMN NAME
         MVC   SQLNAME+2(8),=CL8'_VN'
         AP    VNCT,=P'1'
         UNPK  SQLNAME+5(1),VNCT
         OI    SQLNAME+5,C'Ø'
```

```
         LA    R0,5
         STH   R0,SQLNAME
         LH    R0,SQLNAME
A231     EQU   *
         ST    R0,VL
* SET DATA INTO REXX VARIABLE
         BAL   R14,SETVAR
         LA    R7,SQLVARN_SIZE(R7)
         BCT   R6,A230
* END OF SCAN PHASE, ALLOCATE DATA BUFFER
* R5: TOTAL BUFFER SIZE
         ST    R4,BUFFSIZE
         GETMAIN EU,LV=(4),A=A_DBUF
* COMPLETE SQLDA
         L     R5,A_DBUF            PTR(DATA BUFFER)
         LH    R6,SQLD              # OF OCCURRENCES COLUMNS
         LA    R7,SQLVAR            REINIT POINTER
* COLUMN TYPE
A240     BAL   R14,GET_TYPE
* R3: FIELD SIZE
* R10: DSECT_TYPE ENTRY
         ST    R5,SQLIND            A(INDICATOR), IF NEEDED
         LA    R5,2(R5)             UPDATE PTR
         ST    R5,SQLDATA           A(HOST VARIABLE)
         AR    R5,R3                UPDATE PTR
         LA    R7,SQLVARN_SIZE(R7)
         BCT   R6,A240
* RETRIEVE RECORDS
* DECLARE CURSOR
         EXEC  SQL DECLARE CSR CURSOR FOR S1
         BAL   R14,CHECK_SQL
* OPEN CURSOR
         EXEC  SQL OPEN CSR
         BAL   R14,CHECK_SQL
         ZAP   INDEX,=P'0'         INITIALIZE ROW INDEX
A400     DS 0H
* FETCH ROW
         EXEC  SQL FETCH CSR USING DESCRIPTOR :SQLDA
         CLC   SQLCODE,=F'100'
         BE    EOD                 END OF DATA
         AP    INDEX,=P'1'         INCREMENT INDEX
*  CONVERT INDEX TO FORM: .N
         MVC   VNINDEX,=X'4020202020202120'
         LA    R1,VNINDEX+7
         EDMK  VNINDEX,INDEX
         BCTR  R1,0
         MVI   0(R1),C'.'
         MVC   VNINDEX,0(R1)
         LA    R0,VNINDEX+L'VNINDEX
```

```
         SR    RØ,R1
         ST    RØ,VLINDEX
         LH    R6,SQLD               NO OF OCCURRENCES (COLUMNS)
         LA    R7,SQLVAR             REINITIALIZE POINTER
* WRITE TABLE ROW
A41Ø     DS    ØH
         USING SQLVARN,R7
* DEFAULT (NULL) VALUE
         LA    R2,1
         LA    R3,=C'-'
         L     R1,SQLIND             A(INDICATOR FIELD)
         LH    RØ,Ø(R1)
         CH    RØ,=H'-1'
         BE    A42Ø                  NO DATA
         BAL   R14,GET_TYPE
* R2: DATA WIDTH
* R3: FIELD SIZE
* R1Ø: DSECT_TYPE ENTRY
         MVC   A_DSADDR,DS_ADDR
         L     R15,A_DSADDR          A(ROUTINE)
         BALR  R14,R15
* R3: A(FORMATTED FIELD)
* R2: L(FORMATTED FIELD)
A42Ø     ST    R2,VL
         LH    R1,SQLNAME            L(NAME)
         LA    RØ,L'VN               MAX(L(NAME))
         CR    R1,RØ
         BNH   *+6
         LR    R1,RØ                 TRUNCATE
         BCTR  R1,Ø                  LC(NAME)
         MVC   VN,VN-1
         MVC   VN,SQLNAME+2          COLUMN NAME
         EX    R1,*-6
* SET DATA INTO REXX VARIABLE
         BAL   R14,SETVAR
         LA    R7,SQLVARN_SIZE(R7)
         BCT   R6,A41Ø               GET NEXT COLUMN IN ROW
         B     A4ØØ                  GET NEXT ROW
A5ØØ     DS    ØH                    PROCESS NON-SELECT
         CLC   HOSTSW,YES
         BNE   NOVARS
         EXEC  SQL EXECUTE S1 USING :HOSTVAR
         BAL   R14,CHECK_SQL
         B     EOD
NOVARS   DS    ØH
         EXEC  SQL EXECUTE S1
         BAL   R14,CHECK_SQL
EOD      CALL  DSNALI,(CLOSE,SYNC),VL,MF=(E,CAFCALL)
* END OF PROCESSING
```

```
EOP      DS    ØH
* OUTPUT # OF ROWS PROCESSED
         MVC   WK,=X'FØ202020202020202Ø'
         ED    WK,INDEX
         LA    R3,WK             A(DATA)
         MVC   VL,=A(L'WK)       L(DATA)
         MVC   VLINDEX,=A(Ø)     NO INDEX
         MVC   VN,=C'_NROWS '
* SET DATA INTO REXX VARIABLE
         BAL   R14,SETVAR
* RELEASE ALLOCATED STORAGE
         L     R2,COLSIZE
         LTR   R2,R2
         BZ    NOCOLS
         L     R3,A_SQLDA
         FREEMAIN R,LV=(2),A=(3)
NOCOLS   L     R2,BUFFSIZE
         LTR   R2,R2
         BZ    NOBUF
         L     R3,A_DBUF
         FREEMAIN R,LV=(2),A=(3)
NOBUF    EQU   *
*        DISCONNECT FROM DB2
         CALL  DSNALI,(DISCON),VL,MF=(E,CAFCALL)
GETOUT   EQU   *
         L     R13,4(R13)            RESTORE A(OLD SAVE AREA)
         L     R15,RC                SET RETURN CODE
         EINDE SAVE=SA,RCR=R15
BLOWREXX EQU   *
         MVC   RC,=F'8'
         B     NOBUF
BLOWCON  EQU   *
         MVC   RC,RETCODE
         MVC   CS4BYTE,REASCODE
         BAS   R14,HEXCONV             CONVERT TO EBCDIC
         MVC   REASWORK,CS8BYTE
         LA    R3,REASWORK             A(DATA)
         MVC   VL,=A(L'REASWORK)       L(DATA)
         MVC   VLINDEX,=A(Ø)           NO INDEX
         MVC   VN,=C'_REASON '
* SET DATA INTO REXX VARIABLE
         BAL   R14,SETVAR
         B     NOBUF
BLOW     EQU   *
         MVC   RC,=F'99'
         B     GETOUT
RC       DS    F                       PROGRAM RETURN CODE
*  CAF VARIABLES
HOSTSW   DC    CL1' '
```

```
REASWORK DS    CL8
YES      DC    CL1'Y'
SYNC     DC    CL4'SYNC'
CLOSE    DC    CL12'CLOSE       '
OPEN     DC    CL12'OPEN        '
CONNECT  DC    CL12'CONNECT     '
DISCON   DC    CL12'DISCONNECT  '
RETCODE  DS    F
REASCODE DS    F
RIBPTR   DS    F
SECB     DS    F
TECB     DS    F
SSID     DS    CL4
PLAN     DC    CL8'SQLSPDB '
LIALI    DS    F
CAFCALL  CALL ,(*,*,*,*,*),VL,MF=L
CAFLEN   EQU  *-CAFCALL
*  END CAF VARIABLES
         TITLE 'SUBROUTINES'
*-------------------------------------------------------------------*
*  SUBROUTINE HEXCONV - CONVERTS ADDRESSES TO PRINTABLE HEX EBCDIC   *
*  INPUT: CS4BYTE       OUTPUT: CS8BYTE                              *
*  CSECT MUST BE 240 BYTES OR LONGER                                 *
*-------------------------------------------------------------------*
HEXCONV  DS    ØH
         B     CODING
SAVEHC   DS    18F                    SAVE AREA HEXCONV
CS5BYTE  DS    ØCL5
CS4BYTE  DS    CL4
         DS    C
CS9BYTE  DS    ØCL9
CS8BYTE  DS    CL8
         DS    C
         DC    C'$'
TRTABLE  ORG   *-X'FØ'
         ORG   TRTABLE+X'FØ'
         DC    C'Ø123456789ABCDEF'
CODING   STM   RØ,R15,SAVEHC
         UNPK  CS9BYTE,CS5BYTE
         TR    CS8BYTE,TRTABLE
EXITHC   DS    ØF
         LM    RØ,R15,SAVEHC
         BR    R14
GETVNL   DS    ØH                  DETERMINE ACTUAL LENGTH OF NAME
* INPUT: <VN> - NAME
* OUTPUT: RØ - L(NAME)
* R15 - A(FIRST BLANK)
         LA    R1,L'VN
         SR    RØ,RØ               COUNTER
```

```
         LA    R15,VN
GETVNL1  CLI   Ø(R15),C' '
         BER   R14                    END FOUND
         AH    RØ,=H'1'
         LA    R15,1(R15)
         BCT   R1,GETVNL1
* RØ: L(NAME). WITHOUT TRAILING BLANKS
         BR    R14
         DS    A
SETVAR   ST    R14,SETVAR-4           SET REXX VARIABLE
* <VN>: VAR1ABLE NAME, PREFIX
* <VNINDEX>: VARIABLE NAME, SUFFIX
* <VLINDEX>: LENGTH (VARIABLE NAME, SUFFIX)
* <VL>: L(VARIABLE DATA)
* R3: A(VARIABLE DATA)
         BAL   R14,GETVNL         GET L(VN)
* RØ: L(VN), R15: A(FIRST BLANK IN <VN>)
         MVC   Ø(L'VNINDEX,R15),VNINDEX
         A     RØ,VLINDEX
         LA    R5,IRX_SHVBLOCK
         USING SHVBLOCK,R5
         ST    RØ,SHVNAML
         MVC   SHVNAMA,=A(VN)
         MVI   SHVCODE,SHVSTORE
         ST    R3,SHVVALA
         MVC   SHVVALL,VL
         L     R15,AIRXEXCOM           A(IRXEXCOM)
         CALL  (15),(IRX_IRXEXCOM,Ø,Ø,IRX_SHVBLOCK),VL
         L     R14,SETVAR-4
         BR    R14                     RETURN
GET_TYPE DS    ØH                      GET COLUMN TYPE AND SIZE(S)
* INPUT:
* DSECT_SQLVARN ENTRY
* OUTPUT:
* R2: DATA WIDTH
* R3: FIELD SIZE
* DSECT TYPE ENTRY
         LA    R1Ø,T_TYPE-DS_L
         USING DSECT_TYPE,R1Ø
GETTYPE1 LA    R1Ø,DS_L(R1Ø)
         CLC   SQLTYPE,DS_TYPE
         BNE   GETTYPE1
* ENTRY FOUND, GET COLUMN-LENGTH
         LH    R2,SQLLEN
         LR    R3,R2                   PRESET DATA FIELD SIZE
         CLC   DS_CODE,=CL2'P'        (PACKED) DECIMAL?
         BNE   GETTYPE2                :NO
         SR    R2,R2
         IC    R2,SQLPRCSN             PRECISION
```

```
         LR    R3,R2
         SRL   R3,1                    NO OF DIGIT PAIRS
         LA    R3,1(R3)                TRUE DATA FIELD SIZE
GETTYPE2 CLC   DS_CODE,=CL2'CV'        CHARACTER (VARIABLE)?
         BNE   GETTYPE3               :NO
         LA    R3,2(R3)                ALLOC ROOM FOR LENGTH
GETTYPE3 BR    R14                     RETURN
         TITLE 'CONVERSION ROUTINES'
T_TYPE   DS    ØH                      ALIGNMENT
         DC    H'384',AL1(@CHAR,Ø),CL2'D ',AL4(D_DATE)
         DC    H'385',AL1(@CHAR,Ø),CL2'D ',AL4(D_DATE)
         DC    H'388',AL1(@CHAR,Ø),CL2'T ',AL4(D_TIME)
         DC    H'389',AL1(@CHAR,Ø),CL2'T ',AL4(D_TIME)
         DC    H'392',AL1(@CHAR,Ø),CL2'TS',AL4(D_TIME)
         DC    H'393',AL1(@CHAR,Ø),CL2'TS',AL4(D_TIME)
         DC    H'448',AL1(@CHAR,Ø),CL2'CV',AL4(D_CHARV)
         DC    H'449',AL1(@CHAR,Ø),CL2'CV',AL4(D_CHARV)
         DC    H'452',AL1(@CHAR,Ø),CL2'C ',AL4(D_CHAR)
         DC    H'453',AL1(@CHAR,Ø),CL2'C ',AL4(D_CHAR)
         DC    H'456',AL1(@CHAR,Ø),CL2'CV',AL4(D_CHAR)
         DC    H'457',AL1(@CHAR,Ø),CL2'CV',AL4(D_CHAR)
         DC    H'484',AL1(@NUM,Ø),CL2'P ',AL4(D_DEC)
         DC    H'485',AL1(@NUM,Ø),CL2'P ',AL4(D_DEC)
         DC    H'496',AL1(@NUM,Ø),CL2'I ',AL4(D_INT)
         DC    H'497',AL1(@NUM,Ø),CL2'I ',AL4(D_INT)
         DC    H'5ØØ',AL1(@NUM,Ø),CL2'I ',AL4(D_SIN)
         DC    H'5Ø1',AL1(@NUM,Ø),CL2'I ',AL4(D_SIN)
         DC    H'Ø'                       EOT
D_DATE   EQU   D_CHAR
D_TIME   EQU   D_CHAR
D_CHARV  EQU   D_CHAR
@NULL    EQU   X'Ø1'
@CHAR    EQU   1
@NUM     EQU   2
D_CHAR   DS    ØH                        CHARACTER
         L     R3,SQLDATA                A(DATA)
         LH    R2,SQLLEN                 L(DATA)
         CLC   DS_CODE,=C'CV'
         BNER  R14
         LH    R2,Ø(R3)
         LA    R3,2(R3)
         BR    R14
D_DEC    DS    ØH                        PACKED DECIMAL
         L     R3,SQLDATA                A(DATA)
         SR    R1,R1
         IC    R1,SQLLEN                 L(DATA), PRECISION
* R1: NO. OF DECIMAL DIGITS
         SRL   R1,1
* R1: LENGTH CODE OF PACKED FIELD
```

```
         EX    R1,ZAP
         B     FMT_DEC                 FORMAT DECIMAL FIELD
D_SIN    DS    ØH                      BINARY SMALL INTEGER (2 BYTES)
         L     R3,SQLDATA              A(DATA) IN R3
         MVC   H2(2),Ø(R3)             MAKE SURE ITS HALFWORD BOUNDARY
         LH    RØ,H2
         CNOP  Ø,4
         CVD   RØ,D                    CONVERT IT TO DECIMAL
         B     FMT_DEC                 FORMAT DECIMAL FIELD
D_INT    DS    ØH                      BINARY INTEGER
         L     R3,SQLDATA              A(DATA)
         LH    R1,SQLLEN               L(DATA)
* R1: FIELD SIZE
* CREATE MASK FOR ICM INSTRUCTION
         L     R2,=X'ØØØØØØØF'         LOAD MASK
         SLL   R2,Ø(R1)
         SRL   R2,4
         N     R2,=X'ØØØØØØØF'         R2: ICM MASK
         SR    RØ,RØ
         EX    R2,ICM                  LOAD BINARY VALUE INTO RØ
         CNOP  Ø,4
         CVD   RØ,D                    CONVERT IT TO DECIMAL
         B     FMT_DEC                 FORMAT DECIMAL FIELD
FMT_DEC  DS    ØH                      FORMAT DECIMAL FIELD
* INPUT:
* <D>: PACKED DECIMAL FIELD
* R14: RETURN ADDRESS
* OUTPUT:
* R2: L(FORMATTED FLD)
* R3: A(FORMATTED FLD)
         MVC   EDWK,EDMKINT            MOVE MASK
         LA    R1,EDWK+L'EDWK-2
         EDMK  EDWK,D
* R1: 1ST SIGNIFICANT CHARACTER
         LA    R2,EDWK_E             END ADDR
         SR    R2,R1                 L(FORMATTED FLD)
         LR    R3,R1                 A(FORMATTED FLD)
         BR    R14
* EX INSTRUCTIONS
ZAP      ZAP   D,Ø(Ø,R3)
ICM      ICM   RØ,Ø,Ø(R3)
* WORK FIELDS
SCALE    DS    F
H2       DS    H
FILLER2  DS    ØD
D        DS    PL8
WK       DS    CL8
EDWK     DS    CL17
EDWK_E   EQU   *
```

```
EDMKINT  DC    X'4Ø',13X'2Ø',X'212Ø',C'-'      EDIT MASK INTEGERS
         DS A
CHECK_SQL ST   R14,CHECK_SQL-4     CHECK SQLCODE
         L     R15,SQLCODE
         LTR   R15,R15
         BZR   R14                 :SOL OK
         MVC   RC,SQLCODE
         B     EOD
         TITLE 'DATA AREAS'
STARTWRK DS    ØF
A_DEBUG  DS    A                  VALENTINO
A_SQLDA  DS    A                  A(ALLOCATED SQLDA)
A_DBUF   DS    A                  A(DATA BUFFER)
A_DSADDR DS    AL4
COLSIZE  DC    F'Ø'               COLUMN SIZE
BUFFSIZE DC    F'Ø'               BUFFER SIZE
         DC    C' '               CLEAR BYTE
VN       DS    2CL18              VARIABLE NAME
VL       DS    A                  VARIABLE LENGTH
VNCT     DC    PL4'Ø'             GENERATED NAME COUNT (MAX 9)
INDEX    DC    PL4'Ø'             ROW INDEX
VNINDEX  DS    2CL8
VLINDEX  DS    F
SQL_CA   DS    CL(SQLDLEN)        BASIC SQLCA
SQL_DA   DS    CL16               BASIC SQLDA
         EXEC  SQL INCLUDE SQLCA
         EXEC  SQL INCLUDE SQLDA
SQLVARN_SIZE EQU 44
         LTORG
AIRXEXCOM DS A
IRX_IRXEXCOM DC CL8'IRXEXCOM'
         DS    ØA                 ALIGN
IRX_SHVBLOCK DC (SHVBLEN)X'Ø'
DB2VAR   DC    H'4',CL4' '
         ORG   DB2VAR+2
DB2V     DS    CL4
SELECT   DC    H'8Ø',CL8Ø' '
         ORG   SELECT+2
SQLQUERY DS    CL4Ø96
HOSTVAR  DS    H,CL32760
         ORG   HOSTVAR+2
HOSTVAL  DS    CL32760
         TITLE 'DSECTS'
DSECT_TYPE DSECT
DS_TYPE  DS    HL2
DS_GEN   DS    AL1            GENERIC TYPE (NUMERIC, CHARACTER)
         DS    AL1            FILLER
DS_CODE  DS    CL2
DS_ADDR  DS    AL4
```

```
DS_L     EQU   *-DS_TYPE
         IRXSHVB             DEFINITION OF REXX SHVB
         END
//*
//*            ASSEMBLE IF THE PRECOMPILE RETURN CODE
//*            IS 4 OR LESS
//*
//ASM     EXEC PGM=ASMA90,PARM='OBJECT,NODECK',COND=(4,LT,PC)
//SYSIN    DD  DSN=&&DSNHOUT,DISP=(OLD,DELETE)
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DSN=SYS3.SYMBOLIC,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//SYSLIN   DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(800,(10,10)),DCB=(BLKSIZE=800)
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(800,(10,10),,,ROUND),UNIT=SYSDA
//*
//*            LINKEDIT IF THE PRECOMPILER AND ASSEMBLER
//*            RETURN CODES ARE 4 OR LESS
//*
//LKED    EXEC PGM=IEWL,PARM='XREF,AMODE=31,RMODE=ANY',
//             COND=((4,LT,ASM),(4,LT,PC))
//SYSLIB   DD  DISP=SHR,
//             DSN=DB2T.PB.SDSNLOAD
//         DD  DSN=DB2T.PB.SDSNEXIT,DISP=SHR
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD DD DSN=DB2T.PB.RUNLIB.LOAD(SQLSPDB),
//        DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(1024,(50,50)),UNIT=SYSDA
//SYSIN    DD  *
  INCLUDE SYSLIB(DSNALI)
  NAME SQLSPDB(R)
/*
//*
```

Hans Valentijn
DB2 Systems Programmer
De Nederlandsche Bank NV (The Netherlands)

Display table and column descriptions

The goal of the application programming staff was to have an on-line facility to display DB2 table and column descriptions (name, comment, data type, length, scale, etc). In my opinion the convenient environment for such an application is TSO using ISPF tables and display services. I wrote a small REXX EXEC calling three ISPF panels to display the required data.

DIALOG CONTROL AND DATA FLOW

The dialog function DB2COLS receives control and retrieves the subsystem-id (SSID) from a select service which has the DB2 libraries (SDSNLOAD and RUNLIB.LOAD) already allocated.

At first a selection panel (DB2COLS) for table owner and name (DB2 wildcards are allowed) are displayed. After receiving the search conditions, a DB2 SELECT statement against the SYSIBM.SYSTABLES table is executed in a TSO/DSN session using the DSNTIAUL sample unload table program. Temporary datasets are allocated for the input and output datasets used by the unload program. The SQL result set is stored in a temporary ISPF table named TTAB. Next a table display for panel DB2COLS2 is opened. After entering one or more selection marks, the display process for the column descriptions is initiated. Once again a DB2 SELECT statement against the SYSIBM.SYSCOLUMNS table is executed. The resulting rows are stored in ISPF table CTAB and displayed. Returning (PF3 - End) to the selection screen (DB2COLS) will delete all ISPF tables and datasets.

This process is illustrated in Figure 1.

REXX EXEC DB2COLS

```
/* REXX */
/*=======================================-/
/- Function: Display DB2 table and column descriptions.          -/
/-=======================================-*/
```

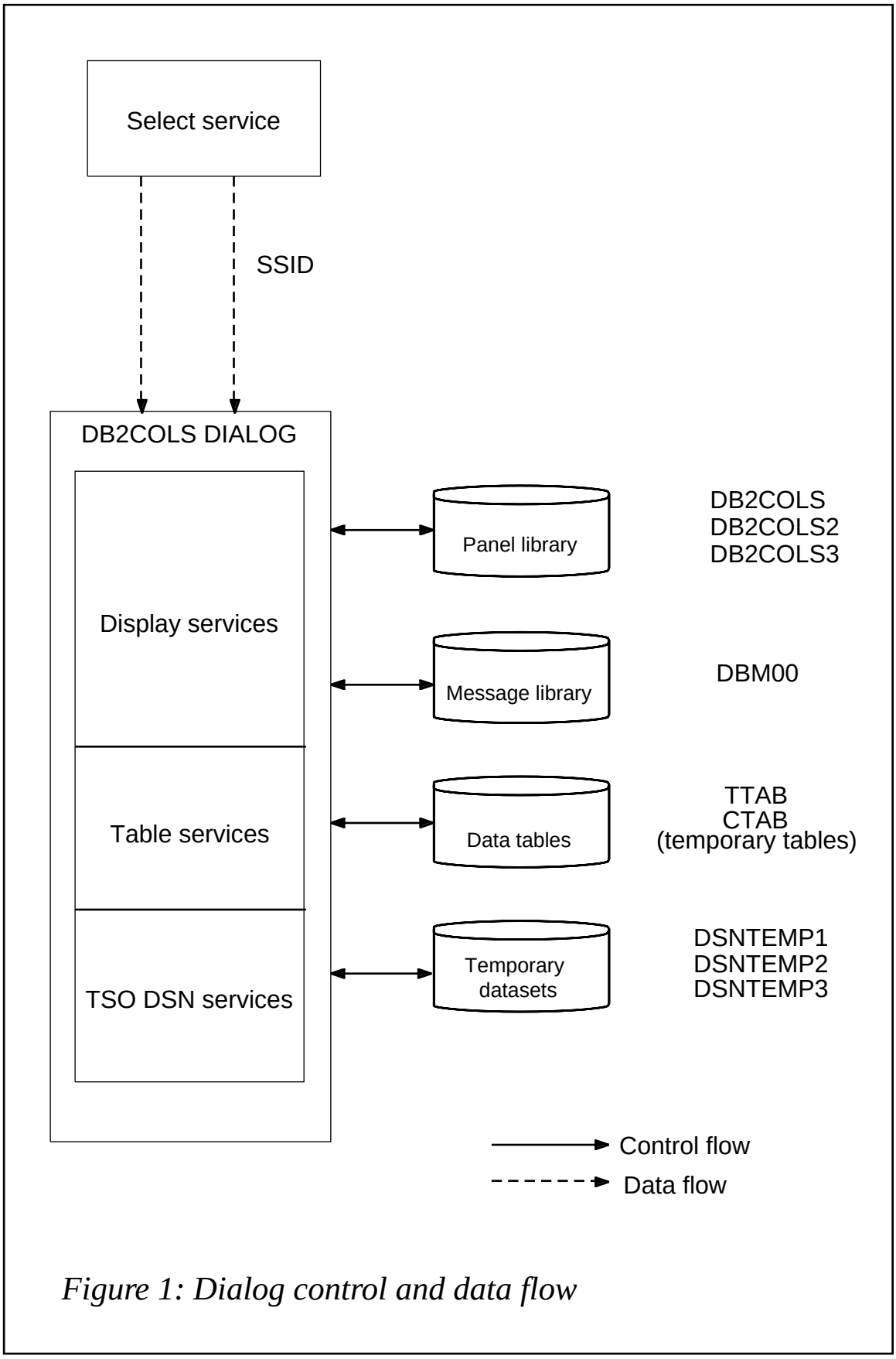

Figure 1: Dialog control and data flow

```
arg dbs .                                  /* parm=DB2 subsystem      */
db2id = dbs                                /*  init panel SSID field  */
select                                            /* init table owner */
  when db2id = 'SSID1' then own = 'OWNER1'
  when db2id = 'SSID2' then own = 'OWNER2'
  when db2id = 'SSID3' then own = 'OWNER3'
  otherwise own = ''
end
tbn     = ''
msg     = ''
amt     = 'PAGE'                                /* init scroll amount */
do forever
    ADDRESS ISPEXEC 'DISPLAY PANEL (DB2COLS)'
    if rc = 8 then
       leave                                                /* pf3 - end  */
    else
     do
       allocrc = Ø                 /* rc of TSO ALLOC cmds           */
       counter = Ø                 /* no of table descriptions displ */
       x = outtrap(var.,'*')       /* trap TSO command output        */
       call alloc                  /* alloc temp datasets for SELECT */
       if allocrc = Ø then
          do
            call db2select1        /* select from SYSTABLES          */
            if counter > Ø then    /* tables for owner found         */
               do
                 ADDRESS ISPEXEC 'TBTOP   TTAB'
                 ADDRESS ISPEXEC 'TBDISPL TTAB PANEL(DB2COLS2)'
                 if rc  < 8 then                          /* tab select */
                    do
                      ADDRESS ISPEXEC 'CONTROL DISPLAY SAVE'
                      call db2select2       /* select from SYSCOLUMNS*/
                      call ctabloop                       /* first row  */
                      call ttabloop                       /* more rows  */
                    end
                 ADDRESS ISPEXEC 'TBEND   TTAB'
                 call dealloc                 /* delete temp datasets */
                 msg = ' ' || counter 'table(s) displayed !'
               end
            else
               do
                 call dealloc                 /* delete temp datasets */
                 msg = ' Table(s) not found !'
               end
          end
     end
end
exit
/*--------------------------------------------------------------------*/
/- subroutine: alloc temp datasets for DB2 select processing          -/
/-             dsntemp1 = SYSIN                                       -/
```

```
/-             dsntemp2 = SYSREC00                                    -/
/-             dsntemp3 = SYSPRINT                                    -/
/-------------------------------------------------------------------*/
alloc:
dsntemp1 = USERID() || '.DB2COLS' || '.A' || TIME('s')
dsntemp2 = USERID() || '.DB2COLS' || '.B' || TIME('s')
dsntemp3 = USERID() || '.DB2COLS' || '.C' || TIME('s')
/*                                          allocate temp dataset 1    */
ADDRESS TSO "ALLOC FI("temp1") DA('"dsntemp1"') OLD REUSE"
if rc = 0 then
   ADDRESS TSO "EXECIO 0 DISKW "temp1" (OPEN FINIS"
else
   ADDRESS TSO "ALLOC FI("temp1") DA('"dsntemp1"'),
   NEW CAT REUSE UNIT(3390),
   LRECL(80) BLKSIZE(27920) RECFM(F B) SPACE(1,1) CYL"
   if rc ¬= 0 then
      do
        msg = 'Temporary dataset 1 cannot be allocated !'
        allocrc = 1
        return
      end
/*                                          allocate temp dataset 2    */
ADDRESS TSO "ALLOC FI("temp2") DA('"dsntemp2"') OLD REUSE"
if rc = 0 then
   ADDRESS TSO "EXECIO 0 DISKW "temp2" (OPEN FINIS"
else
   ADDRESS TSO "ALLOC FI("temp2") DA('"dsntemp2"'),
   NEW CAT REUSE UNIT(3390),
   LRECL(100) BLKSIZE(27900) RECFM(F B) SPACE(1,1) CYL"
   if rc ¬= 0 then
      do
        msg = 'Temporary dataset 2 cannot be allocated !'
        allocrc = 1
        return
      end
/*                                          allocate temp dataset 3    */
ADDRESS TSO "ALLOC FI("temp3") DA('"dsntemp3"') OLD REUSE"
if rc = 0 then
   ADDRESS TSO "EXECIO 0 DISKW "temp3" (OPEN FINIS"
else
   ADDRESS TSO "ALLOC FI("temp3") DA('"dsntemp3"'),
   NEW CAT REUSE UNIT(3390),
   LRECL(133) BLKSIZE(27930) RECFM(F B) SPACE(1,1) CYL"
   if rc ¬= 0 then
      do
        msg = 'Temporary dataset 3 cannot be allocated !'
        allocrc = 1
        return
      end
return
```

```
/*--------------------------------------------------------------------*/
/- subroutine: dealloc and delete temp datasets                       -/
/*--------------------------------------------------------------------*/
dealloc:
ADDRESS TSO "FREE  FI("temp1")"
ADDRESS TSO "FREE  FI("temp2")"
ADDRESS TSO "FREE  FI("temp3")"
ADDRESS TSO "DELETE ('"dsntemp1"')"
ADDRESS TSO "DELETE ('"dsntemp2"')"
ADDRESS TSO "DELETE ('"dsntemp3"')"
return
/*--------------------------------------------------------------------*/
/- subroutine: display table description table                        -/
/*--------------------------------------------------------------------*/
ttabloop:
do forever
   ADDRESS ISPEXEC 'CONTROL DISPLAY RESTORE'
   ADDRESS ISPEXEC 'TBDISPL TTAB'
   select
     when rc = 8 then leave                          /* pf3 - end  */
     when rc = 4 then                                /* more rows  */
        do
          ADDRESS ISPEXEC 'CONTROL DISPLAY SAVE'
          call db2select2
          call ctabloop
        end
     when rc = Ø then                                /* last row   */
        do
          ADDRESS ISPEXEC 'CONTROL DISPLAY SAVE'
          call db2select2
          call ctabloop
        end
   end
end
return
/*--------------------------------------------------------------------*/
/- subroutine: display column description table                       -/
/*--------------------------------------------------------------------*/
ctabloop:
ADDRESS ISPEXEC 'TBTOP   CTAB'
do forever
   ADDRESS ISPEXEC 'TBDISPL CTAB PANEL(DB2COLS3)'
   if rc = 8 then  leave                           /* pf3 - end  */
end
ADDRESS ISPEXEC 'TBEND   CTAB'
return
/*--------------------------------------------------------------------*/
/- subroutine: get table descriptions from SYSIBM.SYSTABLES           -/
/*--------------------------------------------------------------------*/
db2select1:
/*                                        store select into SYSIN */
```

```
V.    = ''
V.1   = "  SELECT                        "
V.2   = "  SUBSTR(NAME, 1, 18),          "
V.3   = "  SUBSTR(REMARKS, 1, 45),       "
V.4   = "  DIGITS(COLCOUNT),             "
V.5   = "  DIGITS(RECLENGTH)             "
V.6   = "  FROM SYSIBM.SYSTABLES         "
V.7   = "  WHERE CREATOR = '" || own || "'"
if tbn = '' then
   V.8   = "  AND TYPE IN('T', 'V')"
else
   V.8   = "  AND TYPE IN('T','V') AND NAME LIKE '" || tbn || "'"
V.9   = "  ORDER BY 1;"
ADDRESS TSO "EXECIO * DISKW "temp1" (STEM V. "
ADDRESS TSO "EXECIO Ø DISKW "temp1" (FINIS"
/*                                                    call DB2 via DSN */
ADDRESS TSO "ALLOC FI("sysin") DA('"dsntemp1"') OLD REUSE"
ADDRESS TSO "ALLOC FI("sysrecØØ") DA('"dsntemp2"') OLD REUSE"
ADDRESS TSO "ALLOC FI("sysprint") DA('"dsntemp3"') OLD REUSE"
ADDRESS TSO "ALLOC FI("syspunch") DUMMY"
push "END"
push "RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB31) PARMS('SQL')"
ADDRESS TSO "DSN SYSTEM(" || dbs || ")"
ADDRESS TSO "EXECIO * DISKR "sysrecØØ" (STEM rowin. FINIS"
ADDRESS TSO "FREE FI("sysin")"
ADDRESS TSO "FREE FI("sysrecØØ")"
ADDRESS TSO "FREE FI("syspunch")"
ADDRESS TSO "FREE FI("sysprint")"
                                        /* create table and add rows */
if rowin.Ø > Ø then
   do
     ADDRESS ISPEXEC 'TBCREATE TTAB KEYS(TNAME),
                      NAMES(REMARKS COL RECL) NOWRITE REPLACE'
     do i=1 to rowin.Ø
        parse var rowin.i tname 19 remarks 64 col 69 recl 74 .
        col  = right(strip(col,'L',Ø),3)
        recl = right(strip(recl,'L',Ø),5)
        ADDRESS ISPEXEC 'TBADD TTAB MULT(' || rowin.Ø || ')'
        counter = counter + 1
     end
   end
return
/*--------------------------------------------------------------------*/
/- subroutine: get column descriptions from SYSIBM.SYSCOLUMNS        -/
/*--------------------------------------------------------------------*/
db2select2:
/*                                          store select into SYSIN */
V.    = ''
V.1   = "  SELECT                        "
V.2   = "  DIGITS(COLNO),                "
V.3   = "  SUBSTR(NAME, 1, 18),          "
```

```
V.4   = "  SUBSTR(REMARKS, 1, 32),      "
V.5   = "  COLTYPE,                    "
V.6   = "  DIGITS(LENGTH),             "
V.7   = "  DIGITS(SCALE),              "
V.8   = "  NULLS,                      "
V.9   = "  DEFAULT                     "
V.1Ø  = "  FROM SYSIBM.SYSCOLUMNS      "
V.11  = "  WHERE TBNAME = '" || tname || "'"
V.12  = "  ORDER BY 1;"
ADDRESS TSO "EXECIO * DISKW "temp1" (STEM V. "
ADDRESS TSO "EXECIO Ø DISKW "temp1" (FINIS"
/*                                                  call DB2 via DSN */
ADDRESS TSO "ALLOC FI("sysin") DA('"dsntemp1"') OLD REUSE"
ADDRESS TSO "ALLOC FI("sysrecØØ") DA('"dsntemp2"') OLD REUSE"
ADDRESS TSO "ALLOC FI("sysprint") DA('"dsntemp3"') OLD REUSE"
ADDRESS TSO "ALLOC FI("syspunch") DUMMY"
push "END"
push "RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB31) PARMS('SQL')"
ADDRESS TSO "DSN SYSTEM(" || dbs || ")"
ADDRESS TSO "EXECIO * DISKR "sysrecØØ" (STEM rowin. FINIS"
ADDRESS TSO "FREE FI("sysin")"
ADDRESS TSO "FREE FI("sysrecØØ")"
ADDRESS TSO "FREE FI("syspunch")"
ADDRESS TSO "FREE FI("sysprint")"
/* create table and add rows */
if rowin.Ø > Ø then
   do
     ADDRESS ISPEXEC 'TBCREATE CTAB,
                      NAMES(CNO COLNAME REMARKS COLTYPE LENG SC N D),
                      NOWRITE REPLACE'
     do i=1 to rowin.Ø
        parse var rowin.i cno 6 colname 24 remarks 56 coltype 64,
                          leng 69 sc 74 n 75 d 76 .
        cno   = right(strip(cno,'L',Ø),3)
        leng  = right(strip(leng,'L',Ø),4)
        sc    = right(strip(sc,'L',Ø),2)
        ADDRESS ISPEXEC 'TBADD CTAB MULT(' || rowin.Ø || ')'
     end
   end
return
```

PANEL DB2COLS

```
)Body
%               DISPLAY TABLE AND COLUMN DESCRIPTIONS   SSID   -+&db2id
%
User   -+&ZUSER
%
Date   -+&date
```

```
% COMMAND ===> _ZCMD
+%Time  -+&ZTIME
%-------------------------------------------------------------------------
+
+
+               %Table owner ..........................:+ _OWN     +
+
+               %Table name (DB2 wildcards allowed) ....:+ _TBN
+
+
+
+
+               &MSG
%-------------------------------------------------------------------------
%
%
)INIT
   .CURSOR = TBN
   &DATE    = '&ZDAY..&ZMONTH..&ZYEAR'
)PROC
   VER (&own,nb,len,'<=',8)
   IF  (&tbn > '')
       VER (&tbn,nb,len,'<=',18)
)END
```

PANEL DB2COLS2

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
)BODY
%
%               DISPLAY TABLE DESCRIPTIONS                    SSID   -
+&db2id
%
User   -+&ZUSER
%
Date   -+&date
%
Time   -+&ZTIME
% COMMAND ===> _ZCMD                                        +%SCROLL
===>_AMT +
%-------------------------------------------------------------------------
% S TNAME              REMARKS                                        COL
RECL
)MODEL
 _Z@tname             @remarks
@col@recl
)INIT
   .ZVARS  = '(SEL)'
   &SEL    = ''
```

```
   .CURSOR = ZCMD
   &DATE   = '&ZDAY..&ZMONTH..&ZYEAR'
)REINIT
   IF   (.MSG = ' ')
        &SEL  = ' '
        REFRESH(SEL)
)PROC
   IF  (&ZTDSELS = Ø)
       .MSG = DBMØØ1
   else
       VER (&SEL,NB,LIST,S)
)END
```

PANEL DB2COLS3

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
)BODY
%
%               DISPLAY COLUMN DESCRIPTIONS               SSID   -
+&db2id
%
User   -+&ZUSER
%
Date   -+&date
%
Time  -+&ZTIME
% COMMAND ===> _ZCMD                                        +%SCROLL
===>_AMT +
%-------------------------------------------------------------------------
% CNO COLNAME            REMARKS                           COLTYPE  LENG
SC N D
)MODEL
 @cno@colname           @remarks                          @coltype
@leng@sc@n@d
)INIT
   .CURSOR = ZCMD
   &DATE   = '&ZDAY..&ZMONTH..&ZYEAR'
)PROC
)END
```

MESSAGE MEMBER DBM00

```
DBMØØ1 'NO SELECTION ENTERED' .TYPE=WARNING
'                             '
```

Raimund Kleebaur
DBA
Hugo Boss (Germany)

Data generator for DB2 – part 3

This month we continue the code for a tool that will generate any kind of test data (with any degree of complexity).

```
LOADSTMT:
      if first = '  ' then load_stmt.lx = load_stmt.lx || ','
      Fld_t    = STRIP(substr(Strucinf.i,26,9))
      lx       = lx + 1 ; ext = 'EXTERNAL'
      if Fld_t = 'SMALLINT' | Fld_t = 'VARCHAR' then do
         ext   =  ; l_l   = ; end
      if Fld_t = 'DATE'      then l_l = '('0010')'
      if Fld_t = 'TIME'      then l_l = '('0008')'
      if Fld_t = 'TIMESTAMP' then l_l = '('0026')'
      if Fld_t = 'INTEGER'   then l_l = '('0011')'
      if Fld_t = 'CHAR' then do
         l_r   = STRIP(substr(Strucinf.i,35,4))   ; ext =
         l_r   = RIGHT(l_r,4,0) ; l_l = '('l_r')' ; end
      if Fld_t = 'DECIMAL' then do
         l_r   = STRIP(substr(Strucinf.i,35,2)) + 2
         l_r   = RIGHT(l_r,4,0) ; l_l = '('l_r')' ; end
      load_stmt.lx = first || LEFT(Field_name.x,20) || 'POSITION(' || ,
           RIGHT(posi,4,0) || ') ' Fld_t ext l_l ; first = '  '
 return
```

DATAMAI2

```
 /********   REXX DATAMAI2 -> DATA-GENERATOR DEFINITION PART   *******/
 /***  General definition section. Please customize to your needs  ***/
 gener_rexx = 'here_your_library' /* Lib, where the gener REXX reside*/
 /* Valid MVS-IDs to connect to a 2-way data sharing DB2 environment */
 v_mvs_id1  = '____' ; v_mvs_id2 = '____'
 userid     = SYSVAR(SYSUID)
 ISPF_Tab   = 'S' || userid      /* Name of ISPF-Table to be created */
 M_pref     = '...'                              /* Message prefix */
 leer       = '                                ' ; nr = '  '
BEGIN:
 do until rc > 0
   "ISPEXEC DISPLAY PANEL(DATADEF1)"     /* Main panel               */
   messtext =                            /* Clean up message text    */
   if D  = 'Y'  then Call STRUDEF0       /* Structure definition ?   */
   if E  = 'Y'  then Call EINZDEF0       /* Field definition ?       */
   if A  = 'Y'  then Call DEFSTORE       /* Generate data ?          */
   if S ¬= ' '  then Call INPUTSAV       /* Save or load structure ? */
 end                                     /* Ok, try it again         */
 "ISPEXEC TBEND "ISPF_Tab""
exit
```

```
STRUDEFØ:                              /* Structure definition section */
 d = 'N' ; def = ' ' ; db2read = 'N'
 "ISPEXEC VGET (ZSYSID)"
 if ZSYSID = v_mvs_id1 | ZSYSID = v_mvs_id2 then do
    "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)" /*Should I read from DB2-Ctlg?*/
    "ISPEXEC DISPLAY PANEL(DATADEF2)"
    "ISPEXEC REMPOP"
    if db2read = 'Y' then do
       "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)" /* Ask for Creator & Table */
       do until rc > Ø ; "ISPEXEC DISPLAY PANEL(DATADEF3)" ; end
       if creator ¬= ' ' & tabelle ¬= ' ' then do
          Call RDDB2DEF CREATOR TABELLE ; db2 = 'X'
       end
       "ISPEXEC REMPOP"
    end
 end
 if topen = 'ON' then Call Strudef2         /* ISPF-Table is open    */
 "ISPEXEC TBOPEN "ISPF_Tab"" "              /* Table ok ?            */
 if rc = 8 then do                          /* No, create it         */
   "ISPEXEC TBCREATE "ISPF_Tab"",
   "NAMES(NR ST FELD DATATYP LANG REST DEF UNI WHAT HOW) KEYS(HF)"
   topen = 'ON'                           /* Now it is ok          */
 end
STRUDEF2:
 if db2 = 'X' then Call DB2DEF
 do until rc > Ø
   "ISPEXEC TBSORT  "ISPF_Tab" FIELDS(HF)"  /* Sort table          */
   "ISPEXEC TBTOP   "ISPF_Tab"" "           /* Set crp to top      */
   "ISPEXEC TBDISPL "ISPF_Tab" PANEL(STRUDEF1)"          /* Show it */
   if choice = 'A' then call ADD_STRU
   if choice = 'D' then call DEL_STRU ; choice =
 end
 SIGNAL BEGIN
ADD_STRU:                                 /* Add structure definition */
 if nt = '' then return; else if nt < 1Ø then nt = 'Ø' || nt /*input?*/
 if sd > Ø then do                        /* Sub-Typ definition ?  */
    hf = 'A' || nt || 'Ø'             /* Build key to get main entry */
    "ISPEXEC TBEXIST "ISPF_Tab"" "          /* Entry ok ?           */
    if rc = Ø then do                       /* Ok, get it           */
       "ISPEXEC TBGET "ISPF_Tab"" "
       if datatyp = 'CHAR' | datatyp = 'VARCHAR' then do
          if def ¬= 'X' then do
             if rest >= lant then do        /* Length ok ?          */
                rest = rest - lant          /* New length           */
                "ISPEXEC TBMOD "ISPF_Tab"" /* Update Main entry     */
             end ; else do                  /* It's too long, sorry  */
                "ISPEXEC SETMSG MSG("M_pref"ØØ4)"
                chk = 'X'                   /* insert not allowed    */
             end
          end
          else do
```

```
            "ISPEXEC SETMSG MSG("M_pref"Ø2Ø)"    /* already defined */
             chk    = 'X'                  /* insert not allowed    */
          end
             /* Mark as SUB-TYP, change datatyp and save this fields */
          feld = '-' || felt ; datatyp = '-SUBDEF-' ; chk1 = 'X'
       end
       ELSE DO
          "ISPEXEC SETMSG MSG("M_pref"ØØ3)"    /* wrong main entry   */
          chk = '' ; chk1 = '' ; sd = ''       /* insert not allowed */
          return
       end
    end
 end
 else sd = Ø
 if datatyt = 'DECIMAL' then do
    "ISPEXEC ADDPOP COLUMN(3Ø) ROW(8)"      /* define window         */
    do until rc > Ø ; "ISPEXEC DISPLAY PANEL(STRUDEF3)" ; end
    "ISPEXEC REMPOP"
    if pt > 18 then pt = 18                 /* check precision       */
    if sc > pt then sc = pt                 /* scale > precision?    */
    if pt < 1Ø then pt = 'Ø' || pt ; if sc < 1Ø then sc = 'Ø' || sc
    if pt <= Ø then lant = 'Ø1,ØØ' ; else lant = pt || ',' || sc
    pt = ; sc =
 end
          /* Let's check, if there is anything to store in the table */
 hf = 'A' || nt || sd ; nr = nt ; st = sd
 if chk1 ¬= 'X' then do                   /* Add structure definition */
    datatyp = datatyt ; feld = felt
 end
                                         /* Init all other fields */
 def = '' ; uni = '' ; what = '' ; how = '' ; lang = lant ; rest = lant
 if chk ¬= 'X' then do                   /* Insert should work    */
    "ISPEXEC TBADD "ISPF_Tab" ORDER"
    if rc = 8 then "ISPEXEC SETMSG MSG("M_pref"ØØ8)" /* Entry exist ?*/
    else lant = ''
 end
 chk = '' ; chk1 = '' ; sd = ''       /* Clean up for the next entry */
 return
DEL_STRU:
 strudel = 'N'
 if nt < 1Ø then nt = 'Ø' || nt             /* Build key             */
 if sd > Ø  then hf = 'A' || nt || sd ; else hf = 'A' || nt || 'Ø'
 "ISPEXEC TBEXIST "ISPF_Tab"" "                          /* Entry ok ? */
 if rc > Ø then "ISPEXEC SETMSG ("M_pref"ØØ6)"          /* No         */
 else do
    chkd = hf                            /* ok, save key          */
    "ISPEXEC TBGET "ISPF_Tab"" "         /* Get it the last time  */
    if sd > Ø then warlang = lang        /* Sub-Typ, save length  */
    if datatyp = 'VARCHAR' | datatyp = 'CHAR' then kannst = 'X'
    "ISPEXEC ADDPOP COLUMN(1Ø) ROW(8)"   /* Define window         */
    "ISPEXEC DISPLAY PANEL(STRUDEF2)"    /* Prompt the user       */
```

```
   "ISPEXEC REMPOP"
   if strudel = 'Y' then do
      "ISPEXEC TBDELETE "ISPF_Tab""        /* Ok, delete            */
      if sd > Ø then do                    /* Sub-Typ ?             */
         hf = 'A' || nt || 'Ø'             /* get main entry        */
         "ISPEXEC TBGET "ISPF_Tab""
         rest = rest + warlang             /* Add length of sub-typ */
         "ISPEXEC TBMOD "ISPF_Tab"" ; sd = ''
      end
      else do
         if kannst = 'X' then do         /* else, find all sub-typs */
            kannst = '' ; cntr = 1
            do while cntr < 1Ø            /* check for sub-typs    */
               hf = 'A' || nt || cntr
               "ISPEXEC TBDELETE "ISPF_Tab""          /* and delete */
               cntr = cntr + 1
            end
            rc = Ø
         end
      end
   end ; chkd = ''                         /* clean up              */
 end
 return
EINZDEFØ:                               /* Field definition section */
 e = 'N'
 if topen ¬= 'ON' then do                /* ISPF-Table ok ?       */
   "ISPEXEC SETMSG MSG("M_pref"Ø18)"    /* No, put it on the screen */
   return
 end
 else do                                 /* Try to open           */
   "ISPEXEC TBQUERY "ISPF_Tab""
   if rc = 8 then do                                  /* Not ok */
      messtext = 'Field definition' ; rc = Ø
      "ISPEXEC SETMSG MSG("M_pref"ØØ9)"
      return                            /* First define a structure */
   end
 end
 "ISPEXEC TBTOP  "ISPF_Tab""             /* Set CRP to top        */
 do until rc > 4
   "ISPEXEC TBDISPL "ISPF_Tab" PANEL(EINZDEF1)" ; save_rc = rc
   do while ztdsels > 1             /* More than one entry selected */
     "ISPEXEC CONTROL DISPLAY SAVE"        /* Save the "old" panel  */
      if choice = 'S' then call DEFINE
      if choice = 'U' then call UNIQUE
     "ISPEXEC CONTROL DISPLAY RESTORE"   /* Restore the "old" panel */
     if ztdsels > 1 then "ISPEXEC TBDISPL "ISPF_Tab""; else ztdsels = Ø
   end
   if ztdsels = 1 then if choice = 'U' then call UNIQUE
   if ztdsels = 1 then if choice = 'S' then call DEFINE
   choice = ' ' ; rc = save_rc     /* Delete choice, and restore RC */
 end
```

```
 SIGNAL BEGIN                                  /* Back to the roots     */
UNIQUE:                              /* Build key to get the main entry */
 nr = nr * 1 ; if nr < 1Ø then nr = 'Ø' || nr ; hf = 'A' || nr || 'Ø'
 "ISPEXEC TBGET "ISPF_Tab""                    /* and get it            */
 univ = uni                                /* Now, check the unique flag */
 if univ = '' then uni = 'Unique yes'          /* off, then on          */
 else uni = ''                                 /* on, then off          */
 "ISPEXEC TBPUT "ISPF_Tab""                    /* save it               */
 return
DEFINE:
 "ISPEXEC TBGET "ISPF_Tab""
 if datatyp = '-SUBDEF-' | datatyp = 'CHAR' | datatyp = 'VARCHAR' then
                               Call CHARDEF1
 if datatyp = 'TIME'      then Call TIMEDEF1
 if datatyp = 'DATE'      then Call DATEDEF1
 if datatyp = 'DECIMAL'   then Call DECIDEF1
 if datatyp = 'INTEGER'   then Call INTEDEF1
 if datatyp = 'SMALLINT'  then Call SMALDEF1
 if datatyp = 'TIMESTAMP' then Call TISTDEF1
 return
CHARDEF1:                             /* Definition of a character field */
 fw  = 'N' ; w  = ' ' ; sw   = 'N' ; sn    = ' '  /* Init all values */
 sv  = ' ' ; sp = ' ' ; so   = ' ' ; ex    = ' '  /*      - " -      */
 sto = ' ' ; le = ' ' ; once = ' ' ; exdsn = ' '  /*      - " -      */
 rep = ' ' ; wb = 'N'                             /*      - " -      */
 v1  = ' ' ; p1 = ' ' ; v2   = ' ' ; p2    = ' '  /*      - " -      */
 v3  = ' ' ; p3 = ' ' ; v4   = ' ' ; p4    = ' '  /*      - " -      */
 v5  = ' ' ; p5 = ' ' ; v6   = ' ' ; p6    = ' '  /*      - " -      */
 "ISPEXEC TBGET "ISPF_Tab""       /* Get the associated table-entry  */
 if lang ¬= rest & datatyp ¬= '-SUBDEF-' then do /*possible to define*/
    "ISPEXEC SETMSG MSG("M_pref"Ø15)"            /* No, forget about */
    return
 end
 if what = '' then           /* Was it defined before ? No, set init */
    how  = '                                      ',
 else do                          /* else, restore the old definition */
    if what = 'F' then do                    /* Fix character string */
       fw   = 'Y' ; w = left(how,lang)
    end
    if what = 'S' then do                          /* Standard values */
       sw   = 'Y' ; sn   = Substr(how,1,1)  ; sv    = Substr(how,2,1)
                    sp   = Substr(how,3,1)  ; so    = Substr(how,4,1)
                    ex   = Substr(how,5,1)  ; exdsn = Substr(how,6,29)
                    sto  = Substr(how,35,2) ; le    = Substr(how,37,2)
                    once = Substr(how,39,1)
    end
    if what = 'W' then do                       /* Value proportions */
       wb   = 'Y' ; v1 = Substr(how,1,lang)  ; p1 = Substr(how,11,2)
                    v2 = Substr(how,13,lang) ; p2 = Substr(how,23,2)
                    v3 = Substr(how,25,lang) ; p3 = Substr(how,35,2)
                    v4 = Substr(how,37,lang) ; p4 = Substr(how,47,2)
```

```
                    v5 = Substr(how,49,lang) ; p5 = Substr(how,59,2)
                    v6 = Substr(how,61,lang) ; p6 = Substr(how,71,2)
   end
 end
 do until rc > Ø
   "ISPEXEC DISPLAY PANEL(CHARDEF1)"
   proz  = Ø
   if  w    ¬= ' '  then w = left(w,lang)
   if  wb    = 'Y' then do
     if v1 ¬= '' then do ; v1 = left(v1,lang) ; proz = p1        ; end
     if v2 ¬= '' then do ; v2 = left(v2,lang) ; proz = proz + p2 ; end
     if v3 ¬= '' then do ; v3 = left(v3,lang) ; proz = proz + p3 ; end
     if v4 ¬= '' then do ; v4 = left(v4,lang) ; proz = proz + p4 ; end
     if v5 ¬= '' then do ; v5 = left(v5,lang) ; proz = proz + p5 ; end
     if v6 ¬= '' then do ; v6 = left(v6,lang) ; proz = proz + p6 ; end
     if proz < 1ØØ then do
        proz = 1ØØ - proz ; "ISPEXEC SETMSG MSG("M_pref"Ø16)" ; end
     if proz > 1ØØ then do
        proz = proz - 1ØØ ; "ISPEXEC SETMSG MSG("M_pref"Ø17)" ; end
   end
 end
 what = ''
 if fw = 'Y' then do                      /* save for type fix value  */
   how = left(w,lang) ; what = 'F' ; end
 if sw = 'Y' then do                      /* save for type std value  */
   how = left(sn,1) || left(sv,1) || left(sp,1) || left(so,1)  || ,
         left(ex,1) || left(exdsn,29)          || left(sto,2) || ,
         left(le,2) || left(once,1)            || left(rep,1)
   what = 'S' ; if ex = 'Y' & exdsn = '' then what = ''
 end
 if wb = 'Y' then do                      /* save for type value prop */
   how = left(v1,1Ø) || left(p1,2) || left(v2,1Ø) || left(p2,2) ,
      || left(v3,1Ø) || left(p3,2) || left(v4,1Ø) || left(p4,2) ,
      || left(v5,1Ø) || left(p5,2) || left(v6,1Ø) || left(p6,2)
   what = 'W'
 end
 Call UPDATE                                 /* Update table entry */
 return
DATEDEF1:                                /* Definition of a date field */
 dateostr = 'ØØØØ31Ø59Ø9Ø12Ø151181212243273304334'
 datemstr = 'ØØØØ31Ø6ØØ911211521822132442743Ø5335'
 stdat = ' ' ; bidat = ' '
 "ISPEXEC TBGET "ISPF_Tab""       /* Get the associated table-entry  */
 if what = '' then how = '                                            '
                                          /* Restore all values */
 dsj = Substr(how,1,4) ; dsm = Substr(how,5,2) ; dst = Substr(how,7,2)
 dej = Substr(how,21,4); dem = Substr(how,25,2); det = Substr(how,27,2)
 did = Substr(how,41,2); ded = Substr(how,43,2)
 "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)"               /* Define window  */
 do until rc > Ø
   datung = ' '
```

```
   "ISPEXEC DISPLAY PANEL(DATEDEF1)"       /* Display window        */
   if dsj ¬= '' then                                /* Startyear ? */
      if dsj = (dsj % 4) * 4 then stdat = 'X'       /* Leap year ? */
   datung =
   if dsm = 2 & dst = 29 & stdat ¬= 'X' then do
      e_y = dsj ; "ISPEXEC SETMSG MSG("M_pref"Ø1Ø)" /* Try it again */
      datung = 'X' ; end
   if dej ¬= '' then                                /* Endyear ?   */
      if dej = (dej % 4) * 4 then bidat = 'X'       /* Leap year ? */
   if dem = 2 & det = 29 & bidat ¬= 'X' then do
      e_y = dej ; "ISPEXEC SETMSG MSG("M_pref"Ø1Ø)" /* Try it again */
      datung = 'X' ; end
   if dsj = dej & dej ¬= ' ' then do         /* Startyear = Endyear */
      sv  = (dsm * 3) - 2 ; svgl = Substr(dateostr,sv,3)
      if stdat = 'X' then   svgl = Substr(datemstr,sv,3)
      svgl = svgl * 1 + dst
      ev  = (dem * 3) - 2 ; evgl = Substr(dateostr,ev,3)
      if bidat = 'X' then   evgl = Substr(datemstr,ev,3)
      evgl = evgl * 1 + det
      if svgl > evgl then                  /* Startdate > Enddate   */
         "ISPEXEC SETMSG MSG("M_pref"Ø12)"          /* Try it again */
   end
 end
 "ISPEXEC REMPOP"
 if datung = ' ' then do
   how = left(dsj,4) || left(dsm,2) || left(dst,2) || left(leer,12) ||,
         left(dej,4) || left(dem,2) || left(det,2) || left(leer,12) ||,
         left(did,2) || left(ded,2)
    Call UPDATE                              /* Update table entry */
 end
 else do
   "ISPEXEC SETMSG MSG("M_pref"Ø19)"   /* Not ok, because of Feb 29  */
   datung =                            /* and year isn't a leap year */
 end
 what = '' ; how = ''                        /* Clean up             */
 return
TIMEDEF1:                          /* Definition of a TIME-field     */
 "ISPEXEC TBGET "ISPF_Tab""        /* Get the associated table-entry */
 if what = ' ' then how = '                                          '
 ths = Substr(how,9,2) ; tms = Substr(how,11,2) ; tss = Substr(how,13,2)
 the = Substr(how,29,2); tme = Substr(how,31,2) ; tse = Substr(how,33,2)
 tit = Substr(how,41,2); tei = Substr(how,43,2)
 "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)"                /* Define window */
 do until rc > Ø
    "ISPEXEC DISPLAY PANEL(TIMEDEF1)"              /* Display panel */
 end
 "ISPEXEC REMPOP"
 how = left(leer,8)  || left(ths,2) || left(tms,2) || left(tss,2) ||,
       left(leer,14) || left(the,2) || left(tme,2) || left(tse,2) ||,
       left(leer,6)  || left(tit,2) || left(tei,2)
 Call UPDATE                                      /* Update table entry */
```

```
 return
TISTDEF1:                          /* Definition of a TIMESTAMP-field */
 "ISPEXEC TBGET "ISPF_Tab""
 how_init = '0001010100000000000009999123123595999999        '
 if what = ' ' then how = how_init
 xj  = Substr(how,1,4)  ; xm = Substr(how,5,2)  ; xt = Substr(how,7,2)
 xh  = Substr(how,9,2)  ; xi = Substr(how,11,2) ; xs = Substr(how,13,2)
 xms = Substr(how,15,6)
 yj  = Substr(how,21,4) ; ym = Substr(how,25,2) ; yt = Substr(how,27,2)
 yh  = Substr(how,29,2) ; yi = Substr(how,31,2) ; ys = Substr(how,33,2)
 yms = Substr(how,35,6)
 zi  = Substr(how,41,6) ; zs = Substr(how,47,2)
 "ISPEXEC ADDPOP COLUMN(14) ROW(5)"
 do until rc > 0
    "ISPEXEC DISPLAY PANEL(TISTDEF1)"
    testd = ' ' ; stdat = ' ' ; bidat = ' '
    if xj ¬= '    '  then                          /* Startyear ? */
       if xj = (xj % 4) * 4 then stdat = 'X' /* Startyear = leap year*/
    if xm = 2 & xt = 29 & stdat ¬= 'X' then do
       dsj = xj ; "ISPEXEC SETMSG MSG("M_pref"01")" ; dsj = ''
    end
    if yj ¬= '    '  then                            /* Endyear ? */
       if yj = (yj % 4) * 4 then bidat = 'X'  /* Endyear = leap year */
    if ym = 2 & yt = 29 & bidat ¬= 'X' then do
       dej = yj ; "ISPEXEC SETMSG MSG("M_pref"01")" ; dej = ''
    end
    if xj = yj then
       if (xm + 1) * 30 + xt > (ym + 1) * 30 + yt then
          "ISPEXEC SETMSG MSG("M_pref"028)"
    if xm = ym & xt = yt then
       if (xh +1) * 3600 + (xi + 1) * 60 + xs > ,
          (yh + 1 * 3600) + (yi + 1) * 60 + xs THEN
          "ISPEXEC SETMSG MSG("M_pref"029)"
 end
 "ISPEXEC REMPOP"
 how_temp = left(xj,4) || left(xm,2) || left(xt,2)  || left(xh,2) ||,
            left(xi,2) || left(xs,2) || left(xms,6) ||,
            left(yj,4) || left(ym,2) || left(yt,2)  || left(yh,2) ||,
            left(yi,2) || left(ys,2) || left(yms,6) ||,
            left(zi,6) || left(zs,2) ; def = ' '
 if how_temp = how_init then how =
 else do ; how = how_temp ; what = 'X' ; def = 'X' ; end
 "ISPEXEC TBPUT "ISPF_Tab""
 what = ' ' ; how = ' '
 return
SMALDEF1:                           /* Definition of a SMALLINT-field */
 "ISPEXEC TBGET "ISPF_Tab""         /* Get the associated table-entry */
 if what = ' ' then how = '              '
 ssi = Substr(how,1,6) ; esi = Substr(how,7,6) ; isi = Substr(how,13,3)
 "ISPEXEC ADDPOP COLUMN(10) ROW(5)"                /* Define window  */
 do until rc > 0                                   /* Display window */
```

```
    "ISPEXEC DISPLAY PANEL(SMALDEF1)"
    if esi ¬= ' ' then
       if ssi > esi then "ISPEXEC SETMSG MSG("M_pref"Ø13)"
    if isi ¬= ' ' then
       if esi - ssi < isi then "ISPEXEC SETMSG MSG("M_pref"Ø14)"
 end
 "ISPEXEC REMPOP"
 how = left(ssi,6) || left(esi,6) || left(isi,3)
 Call UPDATE                                /* Update table entry    */
 return
INTEDEF1:                            /* Definition of an INTEGER-field */
 "ISPEXEC TBGET "ISPF_Tab""
 if what = '' then how = '                          '
 sin = Substr(how,1,11)
 ein = Substr(how,12,11)
 iin = Substr(how,23,4)
 "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)"         /* Define window         */
 do until rc > Ø
    "ISPEXEC DISPLAY PANEL(INTEDEF1)"       /* Display window        */
    if ein ¬= ' ' then                      /* Endvalue set ?        */
       if sin > ein then             /* Yes, startvalue > endvalue ? */
          "ISPEXEC SETMSG MSG("M_pref"Ø13)"        /* This is not ok */
    if iin ¬= ' ' then                      /* Stepvalue set ?       */
       if ein ¬= ' ' & sin ¬= ' ' then
          if sin - ein > iin then
             "ISPEXEC SETMSG MSG("M_pref"Ø14)"     /* This is not ok */
 end
 "ISPEXEC REMPOP"                           /* Remove window         */
 how = left(sin,11) || left(ein,11) || left(iin,11)
 Call UPDATE                                /* Update table entry    */
 return
DECIDEF1:                             /* Definition of a decimal field */
 sdec = Ø ; edec = Ø ; stdec = Ø /* Init start-, end- and step value */
 "ISPEXEC TBGET "ISPF_Tab""        /* Get the associated Table-entry */
 pre = Substr(lang,1,2) ; sca = Substr(lang,4,2)
 pre = pre * 1 ; sca = sca * 1 ; diff = pre - sca
 if what = '' then           /* Was it defined before ? No, set init */
    how = '-Ø                  -Ø                  -Ø                  '
 sdec  = Substr(how,1,2Ø)  ; edec  = Substr(how,21,2Ø)
 stdec = Substr(how,41,2Ø)
 "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)"         /* Define window         */
 NUMERIC DIGITS(pre)
 do until rc > Ø
    "ISPEXEC DISPLAY PANEL(DECIDEF1)" ; Call DECCHECK
 end
 "ISPEXEC REMPOP"
 what = 'D'
 how  = left(sdec,2Ø) || left(edec,2Ø) || left(stdec,2Ø)
 if sdec ¬= '-Ø' | edec ¬= '-Ø' | stdec ¬= '-Ø' then  Call UPDATE
 return
UPDATE:
```

```
 def = 'X'                                        /* Anything defined ? */
 if how  ¬= ' ' & what = ' ' then what = 'X'                 /* Yes */
 if how   = ' ' then def = ' '                               /* No  */
 "ISPEXEC TBPUT "ISPF_Tab""                       /* Update table entry */
 what = ' ' ; how = ' '                           /* Clean up           */
 return
DB2DEF:                    /* Receive the data from the DB2 interface */
 db2 = ' ' ; fromnr = Ø ; not_ok = 'X' ; da = ' ' ; overflow = ' '
 "ISPEXEC TBTOP   TEMP"
 if nr ¬= '  ' then fromnr = nr
 "ISPEXEC ADDPOP COLUMN(1Ø) ROW(5)"
 etext1 = ' Table ' || creator || '.' || tabelle
 etext2 = ' do not exist in DB2'
 do until colno ¬= ' '
    "ISPEXEC TBSKIP TEMP" ; if rc > Ø then SIGNAL DB2DEF2
    "ISPEXEC TBGET TEMP"
    colno = colno * 1 + fromnr
    if colno < 1Ø then do
       nr = 'Ø' || colno ; hf = 'A' || nr || 'Ø' ; end
    else do ; hf = 'A' || colno || 'Ø' ; nr = colno ; end
    if colno > 99 then do
       etext1 = 'Maximum entries for structure reached |'
       etext2 = '     Structure is incomplete !!!       '
       overflow = 'Y' ; Signal DB2DEF2
    end
    st = 'Ø' ; feld = name ; datatyp = coltype
    if datatyp  = 'TIMESTMP' then do
       datatyp  = 'TIMESTAMP' ;  lang = 26 ; end
    if datatyp  = 'TIME'    then lang =  8     /* because the        */
    if datatyp  = 'DATE'    then lang = 1Ø     /* generator do it in */
    if datatyp  = 'INTEGER' then lang = 11     /* EXTERNAL format    */
    if datatyp ¬= 'CHAR'      &   datatyp ¬= 'VARCHAR'   & ,
       datatyp ¬= 'DATE'      &   datatyp ¬= 'TIMESTAMP' & ,
       datatyp ¬= 'TIME'      &   datatyp ¬= 'SMALLINT'  & ,
       datatyp ¬= 'DECIMAL'   &   datatyp ¬= 'INTEGER'   then
       datatyp  = 'ERROR'         /* Any other type is not supported */
    lang = lang ; rest = lang ; def = ; uni = ; what =  ; how  =
    "ISPEXEC TBADD "ISPF_Tab""      /* Put it in the structure table */
    colno = '  ' ; da = 'X' ; datatyp = ' '              /* Clean up */
 end
DB2DEF2:
 "ISPEXEC TBEND TEMP" ; rc = Ø
 if da = ' '| overflow ¬= ' ' then "ISPEXEC DISPLAY PANEL(FEHLDB2)"
 "ISPEXEC REMPOP"
 return
DEFSTORE:
 if adsn  = ' ' then return
 if adsn ¬= savedsn then do
    savedsn = adsn ; checked1 = ' ' ; dsnnr = 1 ; Call DSNCHECK
 end
 if outdsn  = ' '  then return              /* Check OUTPUT-Dataset */
```

```
 if outdsn  = adsn then do
    "ISPEXEC SETMSG MSG("M_pref"Ø21)" ; return ; end
 if outdsn ¬= savedsn2 then do
    savedsn2 = outdsn ; checked2 = ' ' ; dsnnr = 2 ; Call DSNCHECK
 end
 if checked1 = 'Y' & checked2 = 'Y' then Call BUILDJCL
 return
DSNCHECK:
 if dsnnr = 1 then checkdsn = adsn
 if dsnnr = 2 then checkdsn = outdsn
 if dsnnr = 3 then checkdsn = sdsn ; bytes = ' '
 if topen ¬= 'ON' & dsnnr = 1 then do
    messtext = 'Generation' ; "ISPEXEC SETMSG MSG("M_pref"ØØ9)"
    return
 end
 if checkdsn = ' ' then "ISPEXEC SETMSG MSG("M_pref"Ø21)"
 else do
    mem_def = ' ' ; doalloc = ' '                /* Init member-flag */
    if left(checkdsn,1) ¬= "'" then              /* DSN in quotes ?  */
       checkdsn = "'" || userid || '.' || checkdsn || "'"
    if right(checkdsn,2) = ")'" then mem_def = 'Y' /* Member spec. ? */
    available = SYSDSN(checkdsn) ; gendsn = Ø
    if left(available,2) = 'OK'       then gendsn = 1    /* DSN ok ? */
    if left(available,8) = 'MEMBER N' then gendsn = 2    /* DSN ok ? */
    if left(available,8) = 'MEMBER S' then gendsn = 3    /* DSN ok ? */
    q = LISTDSI(checkdsn)                 /* Retrieve DS information */
    if rc > 4 & dsnnr ¬= 2 & gendsn ¬= Ø then do
       rc = Ø ; return
    end
    if gendsn = Ø then
       if dsnnr = 2 then do
          checked2 = 'Y' ; doalloc = 'Y' ; return
       end
       else do                                        /* DS not found */
          if dsnnr = 1 then savedsn = ; else savedsn3 =
          "ISPEXEC SETMSG MSG("M_pref"Ø22)" ; Signal BEGIN
       end
    if SYSDSORG = 'PO' & mem_def = ' ' then do    /* Member not spec.*/
       if dsnnr = 1 then savedsn = ; else savedsn2 =
       "ISPEXEC SETMSG MSG("M_pref"Ø24)" ; Signal BEGIN
    end
    if gendsn = 1 then do
       if dsnnr = 1 & checked1 ¬= 'Y' then do
          "ISPEXEC SETMSG MSG("M_pref"Ø23)" ; checked1 = 'Y'
          Signal BEGIN
       end
       if dsnnr = 2 & checked2 ¬= 'Y' then do
          "ISPEXEC SETMSG MSG("M_pref"Ø23)" ; checked2 = 'Y'
          Signal BEGIN
       end
       if dsnnr = 3 & checked3 ¬= 'Y' & s = 'S' then do
```

```
          "ISPEXEC SETMSG MSG("M_pref"Ø23)" ; checked3 = 'Y'
          Signal BEGIN
       end
    end
    if SYSLRECL < 8Ø & (dsnnr = 1 | dsnnr = 3) then do
       bytes = 8Ø ; "ISPEXEC SETMSG MSG("M_pref"Ø25)"; Signal BEGIN
    end
 end ; gendsn = ' ' ; mem_def = ' '                /* Clean up vars */
 return
BUILDJCL:                                  /* Build JCL for generation */
 jcl. = ; a = 'N' ; acculeng = Ø ; generate = 'X'
 Call COUNT ; Call JCL_VAR ; Call SAVEIT
 "ALLOC FI(OUTJCL) DSN("adsn") SHR"     /* Alloc DS for generate JCL */
 "EXECIO * DISKW OUTJCL (FINIS STEM jcl."           /* Write JCL     */
 "ISPEXEC EDIT DATASET("adsn")"                     /* EDIT the JCL  */
 "FREE  FI(OUTJCL)"                                 /* Free the file */
 return
INPUTSAV:     /* Save, load append or load replace structure section */
 if topen ¬= 'ON' & s = 'S' then return   /* No structure, no action */
 if sdsn   = ' ' then return        /* No DSN, no action, no problem */
 if sdsn  ¬= savedsn3 then do               /* DSN checked before ? */
    savedsn3 = sdsn ; checked3 = ' ' ; dsnnr = 3 ; Call DSNCHECK
 end
 if s = 'S' then do                     /* Save a structure or a part */
    jcl.  = ; generate = ' '
    jcl.1 = 'Save data definition. Structure developed by 'userid
    jcl.2 = 'Records to build..' || right(satz,6)
    jcl.3 = 'Tablespec.........'l|| left(creator2,8) || left(table,18)
    jcl.4 = 'Output dataset....' || left(outdsn,4Ø)
    s = ' ' ; jcntr = 5
    Call SAVEIT                         /* Go and get the table-entries */
    "ALLOC FI(DDIN1) DSN("sdsn") SHR"
    "EXECIO * DISKW DDIN1 (FINIS STEM jcl."
    "ISPEXEC SETMSG MSG("M_pref"Ø27)" ; return
 end
 else do
    in_c = Ø ; fromnr = Ø ; Call READ_IN  /* Go and read the dataset */
    if s = 'L' | s = 'A' then Call GET_IN ; s =
 end
 return
READ_IN:
 "ALLOC FI(DDIN1) DSN("sdsn") SHR"                   /* Allocate DS */
 "EXECIO * DISKR DDIN1 (FINIS STEM input."       /* READ all records */
 "FREE FI(DDIN1)" ; in = 1 ; inputok = ; return       /* Free the DS */
GET_IN:       /* Load (or load append) a saved structure (or a part) */
 what_mess = 'Load' ; if s = 'A' then what_mess= 'Add' /*Load or add?*/
 if s = 'L' & topen = 'ON' then do   /* Load replace and table exist */
    "ISPEXEC TBEND    "ISPF_Tab"" ; topen = ''        /* Drop table */
 end
 if s = 'A' & topen = 'ON' then do                   /* Load append */
    "ISPEXEC TBSORT   "ISPF_Tab" FIELDS(HF)"
```

```
   "ISPEXEC TBBOTTOM "ISPF_Tab""                  /* Get last entry */
   if nr ¬= '' then fromnr = nr
 end
 else do                          /* Create the ISPF-Table, if needed */
   "ISPEXEC TBCREATE "ISPF_Tab"",
   "NAMES(NR ST FELD DATATYP LANG REST DEF UNI WHAT HOW) KEYS(HF)"
   topen = 'ON'
 end
 do in = 1 to input.Ø
   if left(input.in,3) = 'Sav' then iotext = Substr(input.in,23,43)
   if left(input.in,3) = 'Rec' then satz   = Substr(input.in,19,6)
   if left(input.in,3) = 'Tab' then do
      l    = Substr(input.in,19,1)
      if l = 'Y' then do            /* Anything about a LOAD-Stmt ? */
         creator2 = Substr(input.in,2Ø,8)   /* Ok, retrieve creator */
         table    = Substr(input.in,28,18)        /* and table-name */
      end
   end
   if left(input.in,3) = 'Out' then outdsn = Substr(input.in,19,41)
   if left(input.in,1) = 'A'   then do          /* Restore an entry */
      nr = Substr(input.in,5,2)      ; nr = nr + fromnr ; inputok = 'Y'
      if nr < 1Ø then nr = 'Ø' || nr ; in_c = in_c + 1
      if nr > 99 then do                          /* To much entries */
         iostext = what_mess || ' error !!!' ; inputok = 'E'
         iotext  = 'Maximum entries for structure reached |'
      end
      hf = 'A' || nr || Substr(input.in,4,1)
      st      = Substr(input.in,7,1)  ; feld = Substr(input.in,8,18)
      datatyp = Substr(input.in,26,9) ; lang = Substr(input.in,35,4)
      if datatyp = 'DECIMAL' then       lang = Substr(input.in,35,5)
      rest    = Substr(input.in,39,4) ; def  = Substr(input.in,43,1)
      if inputok ¬= 'E' then "ISPEXEC TBADD    "ISPF_Tab""
   end
   if left(input.in,1) = 'W' then do      /* Restore the definition */
      what    = Substr(input.in,2,1)  ; how  = Substr(input.in,3,78)
      if inputok ¬= 'E' then "ISPEXEC TBMOD    "ISPF_Tab""
   end
 end
 if inputok = 'Y' then do                        /* Work is well done */
   iostext = what_mess || ' successful'
   iotext  = in_c||' Entries '||what_mess||'ed from dataset '||checkdsn
 end
 if inputok = '' then do                              /* S**t happens */
   iostext = 'Wrong input'
   iotext  = 'No valid input in dataset ' || checkdsn
 end
 "ISPEXEC SETMSG MSG("M_pref"Ø27)"
 return
COUNT:                            /* Get the length of the structure */
 "ISPEXEC TBSORT "ISPF_Tab" FIELDS(HF)"
```

```
 "ISPEXEC TBTOP  "ISPF_Tab""
 "ISPEXEC TBSKIP "ISPF_Tab""
 do until rc > Ø                       /* Read the complete structure */
    "ISPEXEC TBGET "ISPF_Tab""
    if datatyp ¬= '-SUBDEF-' & datatyp ¬= 'DECIMAL' then do
       acculeng = acculeng + lang
       if datatyp = 'VARCHAR' then acculeng = acculeng + 2
    end
    if datatyp = 'DECIMAL' then acculeng = acculeng + left(lang,2)
    "ISPEXEC TBSKIP "ISPF_Tab""
 end ; return
SAVEIT:                    /* Save structure for reuse or generation */
 "ISPEXEC TBSORT "ISPF_Tab" FIELDS(HF)"
 "ISPEXEC TBTOP  "ISPF_Tab""
 "ISPEXEC TBSKIP "ISPF_Tab""
 do until rc > Ø                       /* Read the complete structure */
    "ISPEXEC TBGET "ISPF_Tab""
    nr = nr * 1 ; if nr < 1Ø then nr = 'Ø' || nr    /* Taylor number */
    langdef = right(lang,4) || right(rest,4)     /* Get length field */
    if datatyp = 'DECIMAL' then langdef = left(lang,8)  /* DECIMAL ? */
    jcl.jcntr = hf   ||   nr    || left(st,1) || left(feld,18) || ,
                left(datatyp,9) || langdef    || left(def,1)   || uni
    jcntr = jcntr + 1
    if def = 'X' then do                   /* Is the entry defined ? */
       jcl.jcntr = 'W' || what || how ; jcntr = jcntr + 1 ; end
    "ISPEXEC TBSKIP "ISPF_Tab""
 end
 if generate = ' ' then do              /* Save only, no generation */
    inputok = 'X' ; iostext = 'Save successful'
    iotext  = 'Structure saved in dataset ' || checkdsn
    "ISPEXEC SETMSG MSG("M_Pref"Ø27)"
 end
 else Call SORT_JCL                     /* Ok, it's a generation ! */
 return
JCL_VAR:             /* Set the appropriate JCL-vars for generation */
 jcl.1  = "Put in your jobcard here "
 jcl.2  = "//EXECTSO EXEC PGM=IKJEFT1A,PARM='DATAGEN1'"
 jcl.3  = "//SYSPROC DD DISP=SHR,DSN="gener_rexx
 jcl.4  = "//SYSTSPRT DD SYSOUT=*                    "
 jcl.5  = "//SYSPRINT DD SYSOUT=*                    " ; jcntr = 7
 if l = 'Y' then
  jcl.6 = "//LOADSTMT DD DISP=SHR,DSN=here_the_dsn_to_put_the_loadstmt"
 else jcntr = jcntr - 1
 jcl.jcntr = "//SORTSTMT DD DISP=(,PASS),UNIT=SYSDA,";jcntr = jcntr + 1
 jcl.jcntr = "//         DCB=(RECFM=FB,BLKSIZE=Ø,LRECL=8Ø),"
 jcntr = jcntr + 1
 jcl.jcntr = "//         SPACE=(TRK,(1,1),RLSE), " ; jcntr = jcntr + 1
 jcl.jcntr = "//         DSN=&&SYSIN             " ; jcntr = jcntr + 1
 jcl.jcntr = "//SYSTSIN  DD DUMMY                " ; jcntr = jcntr + 1
 if doalloc = ' ' then do
```

```
    jcl.jcntr = "//OUTPUT   DD DISP=OLD, " ; jcntr = jcntr + 1 ; end
 if doalloc ¬= ' ' then do
    cy1 = satz * acculeng % 40000 + 1 ; cy2 = cy1 % 10 + 1
    jcl.jcntr = "//OUTPUT   DD DISP=(NEW,CATLG,DELETE), "
    jcntr = jcntr + 1
    jcl.jcntr = "//         DCB=(RECFM=FB,BLKSIZE=0,LRECL="acculeng"),"
    jcntr = jcntr + 1
    jcl.jcntr = "//         SPACE=(CYL,("cy1","cy2"),RLSE),UNIT=SYSDA,"
    jcntr = jcntr + 1 ; checked2 = ' '
 end
 if left(outdsn,1) = "'" then odsn = Substr(outdsn,2,length(outdsn)-1)
 else odsn = userid || '.' || outdsn
 jcl.jcntr = "//         DSN="odsn              ; jcntr = jcntr + 1
 jcl.jcntr = "//STRUCT  DD *                  " ; jcntr = jcntr + 1
 jcl.jcntr = "Records to build.."right(satz,6)  ; jcntr = jcntr + 1
 jcl.jcntr = "Tablespec....."l || left(creator2,8) || left(table,18)
 jcntr = jcntr + 1
return
SORT_JCL:      /* Generate a SORT-Step to sort the generated records */
 jcl.jcntr = "//EXECSORT EXEC PGM=SORT            " ; jcntr = jcntr + 1
 jcl.jcntr = "//SYSOUT  DD SYSOUT=*               " ; jcntr = jcntr + 1
 jcl.jcntr = "//SORTIN  DD DISP=OLD,DSN="odsn       ; jcntr = jcntr + 1
 jcl.jcntr = "//SORTOUT DD DISP=OLD,DSN="odsn       ; jcntr = jcntr + 1
 jcl.jcntr = "//SYSIN   DD DISP=(OLD,DELETE),DSN=&&SYSIN"
return
DECCHECK:
 sf = 0 ; ef = 0 ; stf = 0
 if sdec ¬= '-0' then do
    sf = 0 ; if sdec < 0 then sf = 1 ; sdec  = ABS(sdec)
    sdec_t = TRUNC(sdec)             ; sdec  = sdec - sdec_t
    sdec_t = right(sdec_t,diff)      ; sdec  = sdec_t + sdec
    sdec   = TRUNC(sdec,sca)         ; if sf = 1 then sdec = sdec * -1
 end
 if edec ¬= '-0' then do
    ef = 0 ; if edec < 0 then ef = 1 ; edec  = ABS(edec)
    edec_t = TRUNC(edec)             ; edec  = edec - edec_t
    edec_t = right(edec_t,diff)      ; edec  = edec_t + edec
    edec   = TRUNC(edec,sca)         ; if ef = 1 then edec = edec * -1
 end
 if stdec ¬= '-0' then do
    stf = 0 ; if stdec < 0 then stf = 1 ; stdec  = ABS(stdec)
    stdec_t = TRUNC(stdec)          ; stdec  = stdec - stdec_t
    stdec_t = right(stdec_t,diff)   ; stdec  = stdec_t + stdec
    stdec   = TRUNC(stdec,sca)      ; if stf = 1 then stdec = stdec * -1
 end
 return
```

DATEDEF1

```
PANEL DATEDEF1
```

```
)ATTR
! TYPE(FP)
$ TYPE(NEF) PADC('Ø')  CAPS(ON) JUST(RIGHT)
% TYPE(NEF) PADC(USER) CAPS(ON)
- TYPE(LI)  PADC(USER) CAPS(ON)
# AREA(SCRL)
)BODY WINDOW(5Ø,8) CMD(ZCMD)
!Command ===>%Z                                  !
#SAREA39                                           #
#                                                  #
#                                                  #
#                                                  #
#                                                  #
#                                                  #
)AREA SAREA39
!Field     :-FELD                  !

!Startdate :$Z $Z $Z   ! Enddate  :$Z $Z $Z   !

!Step      :$Z !Unit    :%Z !(DD, MM, YY)
)INIT
.ZVARS = '(ZCMD DST DSM DSJ DET DEM DEJ DID DED)'
.CURSOR = ZCMD
&ZWINTTL = 'DATE-Field definition'
)PROC
IF (&DST ¬= ' ') , IF (&DSM = ' ') , &DSM = 'Ø1'
IF (&DSM ¬= ' ') , VER (&DSM,RANGE,1,12)
IF (&DSM = '1'  OR &DSM = '3' OR &DSM = '5' OR &DSM = '7' OR &DSM = '8')
  VER (&DST,RANGE,1,31)
IF (&DSM = '1Ø' OR &DSM = '12')
  VER (&DST,RANGE,1,31)
IF (&DSM='4'  OR &DSM='6' OR &DSM='9' OR &DSM='11')
  VER (&DST,RANGE,1,3Ø)
IF (&DSM='2')    , VER (&DST,RANGE,1,29)
IF (&DSM ¬= ' ') , IF (&DSJ = ' ') , &DSJ = '19ØØ'
IF (&DSJ ¬= ' ') , VER (&DSJ,RANGE,19ØØ,215Ø)
IF (&DET ¬= ' ') , IF (&DEM = ' ') , &DEM = &DSM
IF (&DEM ¬= ' ') , IF (&DEJ = ' ') , &DEJ = '19ØØ'
IF (&DEM ¬= ' ') , VER (&DEM,RANGE,1,12)
IF (&DEM = '1'  OR &DEM = '3' OR &DEM = '5' OR &DEM = '7' OR &DEM = '8')
  VER (&DET,RANGE,1,31)
IF (&DEM = '1Ø' OR &DEM = '12')
  VER (&DET,RANGE,1,31)
IF (&DEM='4'  OR &DEM='6' OR &DEM='9' OR &DEM='11')
  VER (&DET,RANGE,1,3Ø)
IF (&DEM='2')    , VER (&DET,RANGE,1,29)
IF (&DEM ¬= ' ') , VER (&DEM,RANGE,1,12)
IF (&DEJ ¬= ' ') , IF (&DSJ = ' ')  , &DSJ = '19ØØ'
IF (&DEM ¬= ' ') , IF (&DSM = ' ')  , &DSM = 'Ø1'
IF (&DET ¬= ' ') , IF (&DST = ' ')  , &DST = 'Ø1'
```

```
IF (&DEJ ¬= ' ') , IF (&DSJ ¬= ' ') , VER (&DEJ,RANGE,&DSJ,215Ø)
VER (&DED,LIST,DD,MM,YY)
IF (&DED = 'MM')
  IF (&DID = '12') , &DED = 'YY' , &DID = 'Ø1'
IF (&DED = 'MM') , VER (&DID,RANGE,Ø1,12)
IF (&DED = 'YY') , VER (&DID,RANGE,Ø1,99)
IF (&DED = 'DD') , VER (&DID,RANGE,Ø1,99)
IF (&DED = ' ')
  IF (&DID ¬= ' ') , &DED = 'DD'
)END
```

DECIDEF1

```
PANEL DECIDEF1
)ATTR
! TYPE(FP)
/ TYPE(WT)
$ TYPE(NEF) CAPS(ON)            JUST(RIGHT)
% TYPE(LI)  CAPS(ON) PADC(USER)
# AREA(SCRL)
)BODY WINDOW(35,8) CMD(ZCMD)
!Command ===>$Z                  !
#SAREA39                          #
#                                 #
#                                 #
#                                 #
#                                 #
#                                 #
#                                 #
)AREA SAREA39
 !Field %FELD                !

 !Start :$SDEC                !
 !End   :$EDEC                !
 !Step  :$STDEC               !
 !Decimal =/.! Init-value =/-Ø!
)INIT
.ZVARS = '(ZCMD)'
.CURSOR = SDEC
&ZWINTTL = 'DECIMAL-Field definition'
)PROC
VER (&SDEC ENUM)  , VER (&EDEC ENUM) , VER (&STDEC ENUM)
)END
```

Editor’s note: this article will be concluded next month.

Rainer Cockx
Systems Programmer
R+V Versicherung AG (Germany)

IBM has a worldwide OEM deal with Sapiens, so that the latter can integrate and resell MQSeries and DB2 databases with Sapiens' ObjectPool and Workstation products. The IBM technologies will now, says Sapiens, play an important role in its future enterprise solution.

For further information contact:
Sapiens USA, 2525 Meridian, Parkway, Suite 300, Durham, NC 27713, USA.
Tel: (919) 405 1500.
Sapiens (UK) Ltd, Harman House, 1 George Street, Uxbridge, Middx, UB8 1QQ, UK.
Tel: (01895) 464000.

* * *

Compuware has announced a suite of application and component servers for the AS/400e range, which were apparently developed in association with IBM.

There's the application server, allowing Uniface application components to be deployed on the AS/400, as well as the component server, for integrating software components, such as RPG, with component-based, client/server applications.

But there's also the Uniface OCM driver for sites using IBM's Client Access software, providing fast database access via an SQL interface, and a new Uniface DB2/400 driver, aimed at high-volume, transaction processing applications. The latter product, based on a split driver architecture, allows DB2/400 data files to be accessed through pre-compiled RPG modules, which is designed to result in much better scalability and performance.

For further information contact:
Compuware, 31440 Northwestern Highway, PO Box 9080, Farmington Hills, MI 48334-2564, USA.
Tel: (810) 737 7300.
Compuware, 163 Bath Road, Slough, Berks, SL1 4AA, UK.
Tel: (01753) 774000.

* * *

IBM, through its Global Services unit, has announced three new NT-based offerings, covering integration, BackOffice, and DB2. Integration Services for Windows NT is about migrating from non-NT platforms, including NetWare, and includes a migration from Novell Directory Services to NT server.

There's also Integration Services for Microsoft BackOffice, for design and planning assistance in deploying NT and BackOffice servers. Finally, there's the SmoothStart Services for DB2 Universal Database Enterprise Edition for Windows NT, which is an installation service.

For further information contact your local IBM representative.