In this issue

3  Reformatting DSNTEP2 output
9  Quick table information – part 2
18  End of week administration jobs
32  Plan_table Purge Report
48  DB2 news

© Xephon plc 1999
Contributions
Articles published in DB2 Update are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our Notes for Contributors.

DB2 Update on-line
Code from DB2 Update can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need the user-id shown on your address label.

© Xephon plc 1999. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Reformattting DSNTEP2 output

I have often read something like the following phrase in DB2 Update: ‘If you don’t have a REXX SQL interface you can run your query using DSNTEP2 and massage its output according to your needs...’.

We all know that this suggestion works pretty well for result sets that contain only a few columns; however, it can cost us more of our valuable time if we are dealing with result sets consisting of dozens of columns. The more columns returned by DB2, the more DSNTEP2 output becomes unreadable and hard to process.

The TEP2FILE procedure given here takes DSNTEP2 SYSPRINT output and reformats it to one output line per DB2 result row. It is clear that the limitations of DSNTEP2 also apply for TEP2FILE (length cutting of long VARCHAR columns, no difference between ‘blank’ and ‘Null’, etc). However, compared with DSNTIAUL (DB2 unload program), TEP2FILE offers the following advantages:

• It returns the data in the same way as DSNTEP2 – and that means in a readable way (no internal representation of numbers).

• For further processing using REXX, you have no need to deal with either column positions or VARCHAR length bytes (a simple ‘parse var xyz p1 p2 p3 ...’ is all it takes).

TEP2FILE output comes in two flavours:

• Using the parameter value ‘LIST’ (which is the default), the output consists of fixed-length columns, separated by blanks. Null values or blank columns are substituted with dashes (‘-’).

• Using the parameter value ‘DEL’, the output is written with all columns delimited by semicolons (;). I find this is useful for further processing with a spreadsheet processor.

Whichever parameter value you use, the first line of the TEP2FILE output is always a heading line, taken from the column names provided by DSNTEP2.

TEP2FILE can be executed as a batch TSO step using IKJEFT1A (see the sample JCL below).
TEP2FILE (SAMPLE JCL)

//DB2TEP2 ... your JOB header ...
/// ____________________________ 
/// ** DELETE DATASETS 
/// ____________________________ 
/// IEFBR14 EXEC PGM=IEFBR14 
/// IFILE DD DISP=(MOD,DELETE,DELETE),DSN=dsn.for.dsntep2.output, 
/// DATACLAS=yourdataclass 
/// OFILE DD DISP=(MOD,DELETE,DELETE),DSN=dsn.for.tep2file.output, 
/// DATACLAS=yourdataclass 
/// ____________________________ 
/// ** CALL DSNTEP2 
/// ____________________________ 
/// SQL EXEC PGM=IKJEFT1A 
/// SYSTSIN DD * 
/// DSN SYSTEM (DSN) 
/// RUN PROGRAM (DSNTEP2) PLAN(DSNTEP51) 
/// END 
/// SYSPRINT DD DISP=(*,CATLG,CATLG),DSN=dsn.for.dsntep2.output, 
/// DATACLAS=yourdataclass 
/// SYSTSPRT DD SYSOUT=* 
/// SYSIN DD * 
/// SELECT * 
/// FROM SYSIBM.SYSTABLES 
/// WHERE CREATOR = 'SAPR3'; 
/// ____________________________ 
/// ** CALL TEP2FILE 
/// ____________________________ 
/// TEP2FILE EXEC PGM=IKJEFT1A 
/// SYSPRINT DD SYSOUT=* 
/// SYSTSPRT DD SYSOUT=* 
/// ************* your REXX Library 
/// SYSPROC DD DISP=SHR,DSN=dsn.for.your.rexxlib 
/// IFILE DD DISP=OLD,DSN=dsn.for.dsntep2.output 
/// OFILE DD DISP=(,CATLG,DELETE),DSN=dsn.for.tep2file.output, 
/// DCB=(RECFM=FB,LRECL=4096),DATACLAS=yourdataclass 
/// SYSTSIN DD * 
/// PROF NOPREFIX 
/// TEP2FILE LIST 
/// */ 

TEP2FILE

/* REXX 

TEP2FILE

Take DSNTEP2 output, reformat, and write it to DD OFILE 
(1 Line per row returned from DSNTEP2 Query)
Req'd DD-Cards: IFILE (File with DSNTEP2 output)
OFILE (File for TEP2FILE Output)

Parameters:     LIST (default; fixed Columnwidth; blank Columns are substituted with dashes (-))
DEL (Columns are separated with semicolons)

/*
arg o_parm ;
maxcc = Ø ;
if o_parm = '' then o_parm = 'LIST' ;
retc = check_parms() ;       /* Check parameter */
if retc > Ø then signal TEP2FILE_END ;
retc = read_ifile() ;        /* Read DD IFILE */
if retc > Ø then signal TEP2FILE_END ;
retc = read_rownum() ;       /* Determine number of */
if retc > Ø then signal TEP2FILE_END ;
retc = read_colnames() ;     /* Get names of DB2 cols*/
if retc > Ø then signal TEP2FILE_END ;
retc = init_vars() ;         /* Initialize Array- */
if retc > Ø then signal TEP2FILE_END ;
retc = format_orec() ;       /* Format Outputrecs */
if retc > Ø then signal TEP2FILE_END ;
retc = write_ofile() ;       /* Write DD OFILE */
if retc > Ø then signal TEP2FILE_END ;
TEP2FILE_END:
exit(maxcc) ;                /* End program */

/* ————————————————————————————————————————————————————————————————
Proc: check_parms()
Check Input-Parameter
———————————————————————————————————————————————————————————————— */
check_parms:
if o_parm = '' then o_parm = 'LIST' ;
if o_parm <> 'LIST' & o_parm <> 'DEL' then, do:
say 'Invalid TEP2FILE-Parameter: 'o_parm ;
say 'Valid Parametervalues are : LIST or DEL' ;
say 'Processing stopped' ;
maxcc = 12 ;
return maxcc ;
end;
return(Ø) ;

/* ————————————————————————————————————————————————————————————————
Proc: read_ifile()
Read DSNTEP2 output file
———————————————————————————————————————————————————————————————— */
read_ifile:
"Execio * DISKR IFILE (stem irec. finis)" ;
if rc <> Ø then,
do:	say 'Error Reading Input File (DD IFILE) !!!';
	maxcc = 2Ø;
	return maxcc;
end;
say 'Number of Input-Records: 'irec.Ø;
return(Ø);
/* ————————————————————————————————————————————————————————————————
Proc: write_ofile()
Write TEP2FILE output
———————————————————————————————————————————————————————————————— */
write_ofile:
"Execio * DISKW OFILE (stem orec. finis)";
if rc <> Ø then,
do:	say 'Error Writing Output File (DD OFILE) !!!';
	maxcc = 2Ø;
	return maxcc;
end;
say 'Number of Output-Records: 'orec.Ø '(including Header)';
return(Ø);
/* ————————————————————————————————————————————————————————————————
Proc: read_rownum()
Determine how many rows have been found by DSNTEP2
(number of output records-1)
———————————————————————————————————————————————————————————————— */
read_rownum:
found_row = Ø;
do cc = irec.Ø to 1 by -1 until found_row:
	if index(irec.cc,'_|') > Ø then,
do:
		found_row = 1;
		num_tep2_rows = strip(substr(irec.cc,1,index(irec.cc,'_|')-1),,);
	end;
end cc;
if ¬found_row then,
do:	say 'No rows returned by program DSNTEP2 !!';
	maxcc = 4;
	return(maxcc);
end;
else,	say 'Number of rows returned by program DSNTEP2: 'num_tep2_rows;
return(Ø);
/* ————————————————————————————————————————————————————————————————
Proc: read_colnames()
Determine the name and count of columns in DSNTEP2 result set
This task is accomplished by processing the headings before the
read_colnames:
col_num = Ø ;
col_name. = '' ;
cm = Ø ;
hd_line = '' ;
do cc = 1 to irec.Ø while ( cc-cm < 8Ø ) ;
   if index(irec.cc,' 1_|') > Ø then,
      do;
         cm = cc-2 ;
         hline = strip(irec.cm,,) ;
         if substr(hline,1,1) = '|' then,
            do;
               hd_line = hd_line||hline ;
               hline = translate(hline,' ','|');
               do dd = 1 to words(hline);
                  col_num = col_num + 1 ;
                  col_name.col_num = word(hline,dd) ;
               end;
            end;
         end;
      end;
   end cc ;
   if col_num = Ø then,
      do;
         say 'No Column-Headings found in DSNTEP2-Output!!' ;
         maxcc = 8 ;
         return(maxcc) ;
      end;
   end;
pos_ct = Ø ;
frompos. = '' ;                     /* Array of from positions */
topos. = '' ;                     /* Array of to positions   */
headline. = '' ;
pvar_from = '\ '| ;
pvar_to = ' |';
do cc = 1 to length(hd_line) ;          /* Det. Col positions   */
   if substr(hd_line,cc,2) == pvar_from then,
      do;
         pos_ct = pos_ct + 1 ;
         frompos.pos_ct = cc+1 ;
      end;
   if substr(hd_line,cc,2) == pvar_to then,
      do;
         topos.pos_ct = cc ;
         lendiff = topos.pos_ct - frompos.pos_ct ;
         headline.pos_ct = ,
         strip(substr(hd_line,frompos.pos_ct,lendiff),,);
      end;
   end cc;
pos_ct = pos_ct-1 ;
c1 = length(col_num)+1;
say col_num' Columns in the Result-Set:

do cc = 1 to col_num;
    say '  'substr(cc'.',1,cl)':' col_name.cc;
end cc;
return(Ø);

/* ————————————————————————————————————————————————————————————————
Proc: init_vars()
Initialize array variables that are needed for further processing */
init_vars:
do cc = 1 to num_tep2_rows+1 ;          /* Including header record */
    orecc.cc = '' ;                       /* Array of output records */
exrec.cc = '' ;                       /* Array of work records */
end cc ;
return(Ø);

/* ————————————————————————————————————————————————————————————————
Proc: format_orec()
Concatenate Output-Records and reformat them */
format_orec:
do cc = 1 to irec.Ø ;                              /* Concat rows */
    if index(irec.cc,'_|') > Ø then,                /* Data row !! */
        do;
            vrec = strip(irec.cc,,) ;
            ap = substr(vrec,1,index(vrec,'_|')-1) ;   /* Array pos. */
            if datatype(ap) = 'NUM' then,
                xrec.ap = xrec.ap||,
                substr(vrec,index(vrec,'_|')+1) ;   /* Output(1) */
        end;
end cc ;
if o_parm = 'LIST' then,  
    orecc.1 = ' ' ;
else if o_parm = 'DEL' then orecc.1 = '' ;
do dd = 1 to pos_ct ;
    lendiff = topos.dd-frompos.dd ;
    if o_parm = 'LIST' then,  
        orecc.1 = orecc.1||substr(headline.dd,1,lendiff) ;
    else if o_parm = 'DEL' & dd = 1 then,  
        orecc.1 = orecc.1||substr(headline.dd,1,lendiff),;
    else if o_parm = 'DEL' & dd > 1 then,  
        orecc.1 = orecc.1||';'strip(substr(headline.dd,1,lendiff),);
end dd ;
do cc = 2 to num_tep2_rows+1 ;                     /* Output(2) */
do dd = 1 to pos_ct ;
    vv = cc - 1 ;
    lendiff = topos.dd-frompos.dd ;
    if o_parm = 'LIST' then,  
        do;
            v_char = substr(xrec.vv,frompos.dd,lendiff) ;
Quick table information – part 2

This month we continue the code for the procedure that gives quick DB2 table information, column information, index information, referential integrity information, and generates a report.

TINM2

| Creator : %&tv2 | Reclength : %&tv17 |
| Name : %&tv3 | Keycolumns: %&tv18 |
| Remarks : %&tv4 | |
| Tname : %&tv5 | Status : %&tv19 |
| Dbname : %&tv6 | Checkflag : %&tv20 |
| Obid : %&tv7 | Auditing : %&tv21 |
| Obid : %&tv8 | Createdby : %&tv22 |
TINM3

)Attr Default(%+_

Type( text)    intens(high) caps(on ) color(yellow)
$ Type(output)    intens(high) caps(off) color(yellow)
? Type(text)    intens(high) caps(on ) color(green) hilite(reverse)
# Type(text)    intens(high) caps(off) hilite(reverse)
) Type(text)    intens(high) caps(off) color(white)
[ Type( input)    intens(high) caps(on ) just(left )
] Type( input)    intens(high) caps(on ) just(left ) pad('-'
¬ Type(output)    intens(low ) caps(off) just(asis ) color(green)
$ Type(output)    intens(low ) caps(off) just(asis ) color(yellow)

)Body Expand(/)

%/-/-/- ? Table Column Inquiry +%-/-/-
%Command ===>_zcmd / %Scroll

+Valid cmd:|S+More Information
+Enter Valid cmd and press|Enter+
PFF3+Retum
+Table:$stabinfo +

#cmd#Name    #Coltype #Length#Scale#Nulls#Remarks
+
)Model
]z+¬z ¬z ¬z ¬z ¬z ¬z
)Init
.ZVARS = '(ccmd cname ctyp clen csca cnul crem)'
&amt = PAGE
&ccmd = ''
)Reinit
)Proc
)End
### TINM4

```plaintext
Attr Default(%+_
    | type(text) intens(high) caps(on ) color(green) hilite(reverse)
    # type(text) intens(high) caps(on ) color(yellow) hilite(reverse)
)body window(43,19)
+——— |Detail Column +———
+
+ Name :%&col1
+ Remarks :%&col2
+ Label :%&col3
+ Tbcreator:%&col4
+ Tbname :%&col5
+ Colno :%&col6 + Updates :%&col16
+ Coltype :%&col7 + Default :%&col17
+ Length :%&col8 + Keyseq :%&col18
+ Scale :%&col9 + Fkey :%&col19
+ Nulls :%&col10 + Fldproc :%&col20
+ Colcard :%&col11 + Statst :%&col21
+ High2key :%&col12
+ Low2key :%&col13
+ Default :%&col14
+ value %&col15
+
#PF3 Return+
)init
)proc
)end
```

### TINM5

```plaintext
Attr Default(%+_
    | type(text) intens(high) caps(on ) color(yellow)
$ type(output) intens(high) caps(off) color(yellow)
? type(text) intens(high) caps(on ) color(green) hilite(reverse)
# type(text) intens(high) caps(off) hilite(reverse)
} type(text) intens(high) caps(off) color(white)
[ type( input) intens(high) caps(on ) just(left )
] type( input) intens(high) caps(on ) just(left ) pad(' - ')
¬ type(output) intens(low ) caps( off ) just( asis ) color( green)
$ type(output) intens(low ) caps( off ) just( asis ) color( yellow)
)body Expand(/ /)
%-/-/- ? Table Index Inquiry +%-/-/-
%Command ===> _zcmd                                      /%Scroll
===>_amt +
+Valid cmd: S+More Information
+Enter Valid cmd and press|Enter+
```
|PF3|Return +Table:$tabinfo +
+____________________________________________________________
#Unique#Cluster#Cluster#Cluster +
#Index +
#cmd#Name #Creator # rule # ring # red # ratio # Bpool # type +
)Model )z+¬z ¬z ¬z ¬z ¬z ¬z ¬z ¬z+ 
)Init ).ZVARS = '(icmd iname icre iuni iing ired itio bp ityp)' &amt = PAGE &icmd = '' 
)Reinit )Proc )End

TINM6 
)Attr Default(%) _
| type(text) intens(high) caps(on ) color(green) hilite(reverse) |
< type(text) intens(high) caps(on ) color(yellow) hilite(reverse) |
# type(output) intens(low ) caps(off) just(asis ) color(green) |
$ type(output) intens(low ) caps(off) just(asis ) color(yellow) |
)Body window(40,15) Expand(///) 
)Index Colname +
%Cmd_zcmd +
+Index:$ixinfo +
[PF3 Return +
+____________________________________________________________
#Colname #Ordering# Colcard +
)Model ¬z ¬z ¬z +
)Init ).ZVARS = '(colname ordering colcard)' &amt = PAGE 
)Reinit )Proc )End

TINM7 
)Attr Default(%) _
| type(text) intens(high) caps(on ) color(yellow) |
$ type(output) intens(high) caps(off) color(yellow) |
% Command

PF3+Return $tabl +

# Relationships#Creator #Table

Init

.ZVARS = '(rship rcre rtab reln drule)'

&amt = PAGE

)Reinit

)Proc

)End

TIN00

TINØØ1 .ALARM = YES .WINDOW=NORESP .ALARM = YES

'&message'

PTINF01

* PROCESS GS,OFFSET,OPT(TIME);

PTINF01:PROC(PARMS)OPTIONS(MAIN) REORDER;

******************************************************************************
/* DESCRIPTION: PL/I PROGRAM - SELECTION RESULT */
******************************************************************************

DCL PARMS CHAR(100) VAR;
DCL SYSPRINT FILE STREAM OUTPUT;
DCL NUMSEQ BIN FIXED(31) INIT(Ø);
DCL MCARD PIC'—.—.—.—9';
******************************************************************************
/* DCLGEN TABLE: SYSIBM.SYSTABLES */
******************************************************************************

DCL 1 DCLW,
5 NAME CHAR(18),
5 CREATOR CHAR(8),
5 DBNAME CHAR(8),
5 TNAME CHAR(8),
5 DBID BIN FIXED(15),
5 OBID BIN FIXED(15),
EXEC SQL INCLUDE SQLCA;
IF SUBSTR(PARMS,1,8)=" ' ' THEN CRE=' % ' ;
ELSE DO;
  CALL FUNC(SUBSTR(PARMS,1,8),OUT);
  CRE=OUT;
  IF LENGTH( CRE ) < 8 THEN CRE=CRE||' % ' ;
END;
IF SUBSTR(PARMS,9,18)=" ' ' THEN TAB=' % ' ;
ELSE DO;
  CALL FUNC(SUBSTR(PARMS,9,18),OUT);
  TAB=OUT;
  IF LENGTH( TAB ) < 18 THEN TAB=TAB||' % ' ;
END;

/* SELECTION RESULTS */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
  NAME, CREATOR, DBNAME, TSNAME, DBID, OBID, COLCOUNT

DCL 1 WORKST,
  2 CRE  CHAR(8)  VAR,
  2 TAB  CHAR(18) VAR;

DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
DCL IC  BIN FIXED(15);
DCL OUT  CHAR(18) VAR;

PUT SKIP LIST ('DETAIL TABLE INFORMATION');
CALL FETCH;
DO WHILE (SQLCODE=0):
  NUMSEQ=1;
  MCARD=CARD;
  PUT SKIP LIST ('T '||DBNAME||' '||TSNAME||' '||'
  SUBSTR(NAME,1,18)||' '||CREATOR||' '||MCARD);
  IF EDPROC=' ' THEN EDPROC='--';
  IF VALPROC=' ' THEN VALPROC='--';
  IF CLUSTERTYPE=' ' THEN CLUSTERTYPE='--';
  IF STATUS=' ' THEN STATUS='--';
  IF CHECKFLAG=' ' THEN CHECKFLAG='--';
  IF AUDITING=' ' THEN AUDITING='--';
  IF DATACAPTURE=' ' THEN DATACAPTURE='--';
  PUT SKIP LIST (DBID||' '||OBID||' '||COLCOUNT||' '||EDPROC||' '||
  VALPROC||' '||MCARD||' '||NPAGES||' '||PCTPAGES||' '||
  PARENTS||' '||CHILDREN||' '||RECLENGTH||' '||KEYCOLUMNS
  ||' '||STATUS||' '||CHECKFLAG||' '||AUDITING||' '||CREATEDBY
  ||' '||CREATEDTS||' '||ALTEREDTS||' '||DATACAPTURE||' '||
  PCTROWCOMP||' '||STATTIME||' '||CHECKS||' '||CLUSTERTYPE);
  PUT SKIP LIST (REMARKS);
  CALL FETCH;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
FETCH:PROC;
EXEC SQL FETCH C1 INTO
  , :PARENTS, :CHILDREN, :KEYCOLUMNS, :RECLENGTH, :STATUS
  , :CHECKFLAG, :AUDITING, :CREATEDBY, :CREATEDTS, :ALTEREDTS
  , :DATACAPTURE, :PCTROWCOMP, :STATTIME, :CHECKS, :CLUSTERTYPE;
END FETCH;
FUNC:PROC(INP,OUT):
  DCL INP CHAR(18);
  DCL OUT CHAR(18) VAR;
DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ≠ ' ');
END;
OUT=SUBSTR(INP,1,IC-1);
END FUNC;
END PTINFO1;

PTINFO2

* PROCESS GS,OFFSET,OPT(TIME);
PTINFO2:PROC(PARMS)OPTIONS(MAIN) REORDER;
******************************************************************************
/* DESCRIPTION: PL/I PROGRAM - COLUMN DETAIL 
******************************************************************************
DCL PARMS CHAR(100) VAR;
DCL SYSPRINT FILE STREAM OUTPUT;
DCL 1 WORKST,
  2 CRE CHAR(8) VAR,
  2 TAB CHAR(18) VAR;
******************************************************************************
/* DCLGEN TABLE: SYSIBM.SYSCOLUMNS 
******************************************************************************
DCL 1 DCLW,
  5 NAME CHAR(18) VAR,
  5 COLTYPE CHAR(8),
  5 LENG BIN FIXED(15),
  5 SCALE BIN FIXED(15),
  5 COLNO BIN FIXED(15),
  5 NULLS CHAR(1),
  5 COLCARD BIN FIXED(31),
  5 HIGH2KEY CHAR(8),
  5 LOW2KEY CHAR(8),
  5 UPDATES CHAR(1),
  5 REMARKS CHAR(254) VAR,
  5 DEFAULT CHAR(1),
  5 KEYSEQ BIN FIXED(15),
  5 FOREIGNKEY CHAR(1),
  5 FLDPROC CHAR(1),
  5 LABEL CHAR(30) VAR,
  5 STATSTIME CHAR(26),
  5 DEFAULTVALUE CHAR(254) VAR;
DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
CRE=SUBSTR(PARMS,1,8);
TAB=SUBSTR(PARMS,9,18);
EXEC SQL INCLUDE SQLCA;

/* SELECTION RESULTS */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
   NAME, COLTYPE, LENGTH, SCALE, NULLS, COLCARD, HIGH2KEY
 , LOW2KEY, UPDATES, SUBSTR(REMARKS,1,50), DEFAULT, KEYSEQ
 , FOREIGNKEY, FLDPROC, LABEL, DATE(STATSTIME), DEFAULTVALUE, COLNO
FROM SYSDBM.SYSCOLUMNS
WHERE TBCREATOR = :CRE
   AND TBNAME    = :TAB
ORDER BY COLNO
FOR FETCH ONLY;

EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO

DO WHILE (SQLCODE=0);
   PUT SKIP LIST ('C '||NAME||' '||COLTYPE||' '||LENG||' '||SCALE||' '||NULLS||' '||REMARKS);
   IF LABEL       =' ' THEN LABEL       ='-';
   IF DEFAULTVALUE=' ' THEN DEFAULTVALUE='-';
   IF HIGH2KEY    =' ' THEN HIGH2KEY    ='-';
   IF LOW2KEY     =' ' THEN LOW2KEY     ='-';
   IF UPDATES     =' ' THEN UPDATES     ='-';
   IF REMARKS     =' ' THEN REMARKS     ='-';
   IF DEFAULT     =' ' THEN DEFAULT     ='-';
   IF FOREIGNKEY  =' ' THEN FOREIGNKEY  ='-';
   IF FLDPROC     =' ' THEN FLDPROC     ='-';
   PUT SKIP LIST ('D '||COLCARD||' '||COLNO||' '||HIGH2KEY||' '||LOW2KEY||' '||UPDATES||' '||DEFAULT||' '||KEYSEQ||' '||FOREIGNKEY||' '||FLDPROC||' '||STATSTIME||' '||DEFAULTVALUE);
   PUT SKIP LIST ('L '||LABEL);
   EXEC SQL FETCH C1 INTO
END;

EXEC SQL CLOSE C1;
END PTINFO2;

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

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia) © Xephon 1999
End of week administration jobs

INTRODUCTION
There are many tasks that the DBA should execute regularly. Most DBAs concentrate on creating regular back-ups (daily, weekly, monthly, etc). Other tasks are usually done on an ad hoc basis or in ‘accident’ situations, i.e. reorganizations, loads, RUNSTATS, recoveries, etc. However, when you look at these utilities, some could be executed regularly, such as RUNSTATS and modify recovery deletes. So what is there to prevent DBAs from executing them regularly – and are there other specific DB2 commands that could also be executed regularly?

Firstly, let’s briefly look at the utilities and commands covered in this article. For more information look in the appropriate DB2 manuals.

RUNSTATS
RUNSTATS is a DB2 utility used mainly for the purpose of informing somebody of the values contained in DB2 tables and indexes, and the overall ‘health’ of those objects. That ‘somebody’ could be the DB2 optimizer or DBA. Based on the RUNSTATS information, DB2 will know how to process SQL statements in order to process user requests with the fewest resources and in the quickest way. Also, the DBA will know whether there is a need to plan the reorganization of some of the objects that are not in such good shape (lower cluster ratio, number of relocated rows, etc).

Most DBAs know that RUNSTATS should be run after loading, reorganization, recoveries, creation of the new indexes, etc. Most of them execute RUNSTATS after those actions because they are the ones who execute them. But are they aware of the changes in the data in the tables and indexes (extensive inserts, deletes, updates)?

In the process of designing tables and indexes, an estimate is usually made of how the data will change (number of inserts, updates and deletes, allowed values, value distribution, etc), and the DBA is
informed of these estimates. But let’s face it, the world changes and DBAs are not always informed.

Sometimes users start to order some specific item, and the number of those items becomes significant considering the total number of all items. It’s also the custom that programmers change data without making others aware of this (change of codes, using ‘01-01-0001’ instead of null for an unknown date, etc).

And who is blamed when a user starts to yell that something worked quickly before and is now slow? – the programmer says “I made the change two months ago so if the problem hasn’t arisen before surely I can’t be responsible now!”

If the DBA is aware that there are relevant data changes in specific tables, he creates the RUNSTATS job for that table that is executed regularly. But there is always the problem of maintaining it in order to add new objects, delete old objects, etc.

MODIFY RECOVERY DELETE

MODIFY RECOVERY DELETE is a DB2 utility used for the purpose of deleting outdated information from the DB2 catalog relating to recovery information.

Is there really a need to keep information about image copies from two years ago when you don’t have those copies any more? Even though this is a small amount of information for one tablespace for one day or image copy, if you multiply that number by the number of days, number of image copies, frequency of image copies, number of tablespaces, etc, the system catalog will be unnecessarily overloaded with information that doesn’t mean anything to anybody.

REBIND PLAN AND REBIND PACKAGE

REBIND PLAN and REBIND PACKAGE are the DB2 commands used in the case of changes that affect plans or packages, but there is no change in SQL statements. The only way to inform the DB2 optimizer about new statistics collected by RUNSTATS is by using REBIND PLAN and REBIND PACKAGE commands. The same
could be also done by using the BIND PLAN and BIND PACKAGE commands, but then the parameters that affect the plan and package execution should also be specified.

THE PROBLEM
RUNSTATS, REBIND PLAN, REBIND PACKAGE, and MODIFY RECOVERY DELETE are the activities that should be executed regularly against objects in DB2 subsystems, for the reasons explained above.

But those DB2 utilities and commands do not allow the ‘famous’ asterisk (*) in order to execute them against all allowed objects, so the DBA is forced to tailor and maintain jobs in which all objects will be listed.

THE SOLUTION
The operator regularly submits jobs during off-peak hours, one after another – at my location once a week, at the weekend, in order to execute RUNSTATS, REBIND PLAN, REBIND PACKAGE, and MODIFY RECOVERY DELETE against objects in a specific DB2 subsystem. Once prepared and tested, those jobs don’t have to be maintained any more, because they will be executed against all existing objects (tablespaces, plans, and packages).

For each subsystem and each utility or command that should be executed, the DBA should create a separate job. All those jobs are parsing the DB2 SSID and utility/command name to the one, previously customized, REXX EXEC.

It is important to implement the policy that RUNSTATS is executed before REBIND PLAN and REBIND PACKAGE administration jobs.

EOWXXY
EOWXXY should be saved under a separate name for RUNSTATS and MODIFY RECOVERY DELETE utilities for each DB2 subsystem that is to be administered. The name of the job should contain the
utility name abbreviation (XX) and DB2 SSID (Y) so they can be easily distinguished.

Customize the job according to your environment. Here is the code:

```plaintext
//EOWXXY JOB 'JOIVANC','JOIVANC'.
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440,
// NOTIFY=JOIVANC
//
//* NAME        : END OF WEEK ADMINISTRATION JOB
//*
//* DESCRIPTION : THIS JOB IS A MODEL FOR THE END OF WEEK
//* ADMINISTRATION OF DB2 SUBSYSTEM
//* UTILITY SPECIFIED WILL BE EXECUTED AGAINST
//* ALL DB2 OBJECTS IN THE SUBSYSTEM, THEREFORE
//* THIS JOB SHOULD BE EXECUTED ONLY DURING
//* OFF-PEAK HOURS.
//*
//* XX SPECIFIES WHAT UTILITY IS EXECUTED:
//*
//* RS  RUNSTATS
//* MR  MODIFY RECOVERY DELETE
//*
//* Y  SPECIFIES AGAINST WHICH DB2 SUBSYSTEM
//* THE UTILITY IS EXECUTED, IE
//*
//* T  TEST  DB2 SUBSYSTEM (SSID DSNT)
//* P  PRODUCTION DB2 SUBSYSTEM (SSID DSNP)
//*
//* SPECIFY ALSO THE DATASET NAMES ACCORDING TO YOUR INSTALLATION
//*
//* JOBLIB    CREATED DURING DB2 INSTALLATION
//* SYSEXEC   PDS WHERE THE CALLING EXEC (EOWEXEC) RESIDES
//*
//* JOBLIB    DD DISP=SHR,DSN=DSN310.SDSNLOAD
//* SYSDB2P1  EXEC PGM=IKJEFT01,DYNAMNBR=30
//* SYSEXEC  DD DISP=SHR,DSN=DB2.EOW.EXEC
//* SYSTSPRT DD SYSPUT=*%EOWEXEC
//* JCL      DD DSN=&&JCL,DISP=(MOD,PASS),UNIT=SYSDA,
//*           DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//* SYSTSIN  DD *
%EOWEXEC
*********************************************************************
** SPECIFY THE EXEC PARAMETERS:
**
** UTILITY
** ALLOWED VALUES : RUNSTATS
** MODIFYRECOVERYDELETE
** DEFAULT VALUE  : <NONE>
** NOTED IN JOB AS: UTILITY_NAME
```
** SYSTEM
** ALLOWED VALUES: <DEPENDS ON INSTALLATION>
** DEFAULT VALUE : DSN
** NOTED IN JOB AS: DB2_SUBSYSTEM_ID
*********************************************************************
UTILITY : UTILITY_NAME
SYSTEM : DB2_SUBSYSTEM_ID
/*
//DSNUPROC  EXEC DSNUPROC,
//          SYSTEM=DB2_SUBSYSTEM_ID,
//          LIB='DSN310.SDSNLOAD',
//          SIZE=4M,
//          UID='EOWXXY',
//          UTPROC=''
//SYSPRINT DD SYSOUT=* 
//SYSIN DD DSN=&&JCL,DISP=(OLD,DELETE)
//   IF (RC LE 4) THEN
//WTOOK EXEC WTO
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
*********************************************************************
* SSID: DB2_SUBSYSTEM_ID
* END OF WEEK UTILITY_NAME SUCCESSFULL
*********************************************************************
/*
// ELSE
//WTOERR EXEC WTO
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
*********************************************************************
* ERROR
* SSID: DB2_SUBSYSTEM_ID
* END OF WEEK UTILITY_NAME ERROR
*********************************************************************
/*
// ENDIF

EOWRXY
EOWRXY should be saved under a separate name for REBIND PLAN and REBIND PACKAGE commands for each DB2 subsystem that is administered. The name of the job should contain a command name abbreviation (X) and DB2 SSID (Y) so they can be easily distinguished.

Customize the job according to your environment. Here is the code:

//EOWRXY JOB 'JOIVANC','JOIVANC',

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440, 
// NOTIFY=JOIVANC
//*
//** NAME : END OF WEEK ADMINISTRATION REBIND JOB
//**
//** DESCRIPTION : THIS JOB IS A MODEL FOR THE END OF WEEK
//** PLAN AND PACKAGE ADMINISTRATION OF DB2 SUBSYSTEM
//** COMMAND SPECIFIED WILL BE EXECUTED AGAINST
//** ALL DB2 OBJECTS IN THE SUBSYSTEM, THEREFORE
//** THIS JOB SHOULD BE EXECUTED ONLY DURING
//** OFF-PEAK HOURS AND AFTER RUNSTATS.
//**
//** X SPECIFIES WHAT COMMAND IS EXECUTED:
//**
//** P REBIND PLAN
//** K REBIND PACKAGE
//**
//** Y SPECIFIES AGAINST WHICH DB2 SUBSYSTEM
//** THE COMMAND IS EXECUTED, IE
//**
//** T TEST DB2 SUBSYSTEM (SSID DSNT)
//** P PRODUCTION DB2 SUBSYSTEM (SSID DSNP)
//**
//** SPECIFY ALSO THE DATASET NAMES ACCORDING TO YOUR INSTALLATION
//**
//** JOBLIB CREATED DURING DB2 INSTALLATION
//** SYSEXEC PDS WHERE THE CALLING EXEC (EOWEXEC) RESIDES
//**
//** JOBLIB DD DISP=SHR,DSN=DSN310.SDSNLOAD
//** SYSEXEC DD DISP=SHR,DSN=DB2.EOW.EXEC
//** SYSTSPRT DD SYSOUT=* 
//** JCL DD DSN=&JCL,DISP=(MOD,PASS),UNIT=SYSDA,
//** DCB=(RECFM=FB,LRECL=80,BLKSIZE=800) 
//** SYSTSIN DD *
%
%EOWEXEC
*********************************************************************
** SPECIFY THE EXEC PARAMETERS:
**
** COMMAND
** ALLOWED VALUES : REBINDPLAN
** REBINDPACKAGE
** DEFAULT VALUES : <NONE>
** NOTED IN JOB AS: COMMAND_NAME
**
** SYSTEM
** ALLOWED VALUES: <DEPENDS ON INSTALLATION>
** DEFAULT VALUES: DSN
** NOTED IN JOB AS: DB2_SUBSYSTEM_ID
*********************************************************************
COMMAND : COMMAND_NAME
SYSTEM : DB2_SUBSYSTEM_ID

/*
// IF (RC LE 4) THEN
//REBIND EXEC PGM=IEFBR14 ,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSUDUMP DD DUMMY
//SYSIN DD DUMMY
//SYSTSIN DD DSN=&&JCL,DISP=(OLD,DELETE)
// ENDIF
//*/

// IF (RC LE 4) THEN
//WTOOK EXEC WTO
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
*****************************************************************************
* SSID: DB2_SUBSYSTEM_ID
* END OF WEEK COMMAND_NAME SUCCESFULL
*****************************************************************************
/*
// ELSE
//WTOERR EXEC WTO
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
*****************************************************************************
* ERROR
* SSID: DB2_SUBSYSTEM_ID
* END OF WEEK COMMAND_NAME ERROR
*****************************************************************************
/*
// ENDIF

EOWEXEC

Save the REXX EXEC EOWEXEC into PDS, as noted by the SYSEXEC DD statement of the jobs above.

Customize the EXEC according to your installation (SSIDs, libraries, etc). Also customize whether you want the REBINDs to be enabled. The SQL SELECT statements could also be modified if some objects are to be excluded from administration. The administration statements could also be modified, ie change EXPLAIN(YES) into EXPLAIN(NO), change the age parameter, etc.

Here is the code:
/* REXX ********************************************************************/
/* EXEC : EOWEXEC                                                        */
/* EXEC for tailoring SYSIN DD for end of week administration job        */
/* Authority: RACF                                                       */
/* DB2                                                                  */
/* Requirements : If invoked in the TSO/E address space,                 */
/* SYSEXEC DD must be assigned to the dataset                            */
/* in which this EXEC resides.                                          */
/* Changed :                                                            */
/* ***********************************************************************

/* Environment assignent                                                 */
NUMERIC DIGITS 4;

/* Constants:                                                            */
const.inp='SYSIN';            /* DDNAME for TSO DSN input                                           */
const.out='SYSPRINT';         /* DDNAME for TSO DSN output                                            */
const.jcl='JCL';              /* DDNAME for tailored output                                           */
const.rebind='YES';           /* Allow Rebind                                                          */
const.defssid='DSN';          /* Default SSID                                                           */
const.defutil='';             /* Default Utility name                                                  */
const.prog='DSNTEP2';         /* PROGRAM name                                                          */
const.plan='DSNTEP31';        /* PLAN name                                                             */

/* SSID'S Dependent Constants (take care of the order):                  */
const.ssids='DSNT DSNP';      /* SSIDs                                                                 */
const.libs='DSNT31Ø.RUNLIB.LOAD DSNP31Ø.RUNLIB.LOAD';  /* Libraries */

const.runstats="SELECT DBNAME, NAME",
                  "FROM SYSIMB.SYSTABLESPACE",
                  "WHERE DBNAME NOT IN('DSNDBØ1','DSNDBØ7')",
                  "ORDER BY 1, 2";
const.rebindplan="SELECT NAME",
                  "FROM SYSIMB.SYSPLAN",
                  "ORDER BY 1";
const.rebindpackage="SELECT LOCATION CONCAT'.CONCAT COLLID CONCAT'."",
                    "CONCAT NAME CONCAT'.' CONCAT VERSION CONCAT'')",
                    "FROM SYSIMB.SYSPACKAGE",
                    "ORDER BY 1";
const.modifyrecoverydelete="SELECT DBNAME, NAME",
                          "FROM SYSIMB.SYSTABLESPACE",
                          "WHERE DBNAME NOT IN('DSNDBØ7')",
                          "ORDER BY 1, 2";
/* Variables: */
stat. =''; /* SQL STATEMENT words */
dsnerr='no'; /* DSN processing error */
inpl. =''; /* Input lines from SYSPRINT */
inpn =Ø; /* Input lines number */
obl. =''; /* Object list */
obn =Ø; /* Object number */
outl. =''; /* Output lines for Object list */
outn =Ø; /* Output lines number */
parm. =''; /* Input parameters */

SAY '':
SAY '* Processing started..... '*:
SAY '* Date: '||date()||' Time: '||time();
SAY '':

SAY '':
SAY '* Assigning default values...... '*:
SAY '':

tmpstr=translate(inpstr,' ','.');
SELECT;
WHEN word(tmpstr,1)='SYSTEM' THEN parm.ssid=word(tmpstr,2);
WHEN word(tmpstr,1)='UTILITY' THEN parm.util=word(tmpstr,2);
WHEN word(tmpstr,1)='COMMAND' THEN parm.util=word(tmpstr,2);
OTHERWISE CALL expr(1Ø);
END;

lpos=wordpos(parm.ssid, const.ssids);
SELECT;
WHEN lpos=Ø THEN CALL expr(3);
OTHERWISE DO;
   parm.lib=word(const.libs,lpos);
   IF parm.lib='' THEN CALL expr(4);
END;
END;

SELECT
WHEN parm.util='RUNSTATS' THEN parm.stat=const.runstats;
WHEN parm.util='REBINDPLAN' THEN parm.stat=const.rebindplan;
WHEN parm.util='REBINDPACKAGE' THEN parm.stat=const.rebindpackage;
WHEN parm.util='MODIFYRECOVERYDELETE' THEN
   parm.stat=const.modifyrecoverydelete;
OTHERWISE CALL expr(9);
END;
IF const.rebind<>'YES' & pos('REBIND',parm.util)<>)Ø THEN CALL expr(11):

SAY '';
SAY '* Listing current values......*';
SAY '':
SAY ' SYSTEM :'parm.ssid;
SAY ' UTILITY/COMMAND :'parm.util;
SAY ' LIBRARY :'parm.lib;
SAY ' SQL STATEMENT :'parm.stat;

SAY '':
SAY '* Allocating files*':
SAY '':
"ALLOC FI("const.inp")SPACE(1,1) TRACK LRECL(ØØ)RECFM(F B)BLKSIZE(312Ø), REUSE";
IF rc<>Ø THEN CALL expr(1);
"ALLOC FI("const.out")SPACE(1,1) TRACK LRECL(ØØ)RECFM(F B)BLKSIZE(312Ø), REUSE";
IF rc<>Ø THEN CALL expr(2);

SAY '':
SAY '* Executing SQL SELECT STATEMENT*':
SAY '':
DO i=1 TO words(parm.stat);
   parm.stat.i=word(parm.stat,i);
END;
"EXECIO * DISKW "const.inp" (STEM parm.stat. FINIS)"
QUEUE "RUN PROGRAM("const.prog") PLAN("const.plan") LIB(''parm.lib'')":
QUEUE "END";
"DSN SYSTEM("parm.ssid")":
IF rc<>Ø THEN dsnerr='yes';

/* Uncomment the following statements if you want to browse SYSPRINT */
/* Uncomment ONLY if procedure executed from ISPF */
/* brdd = const.out; 
   ADDRESS ISPEXEC "LMINIT DATAID(ddvar) DDNAME("brdd") " 
   ADDRESS ISPEXEC "BROWSE DATAID("ddvar") "; 
   ADDRESS ISPEXEC "LMFREE DATAID("ddvar") "; 
*/

IF dsnerr='yes' THEN DO;
   DO i=1 TO queued();
      PULL stackitem;
   END;
   CALL expr(5);
END;

SAY ' ';
SAY '* Reading SYSPRINT as result of SQL SELECT statement        *';
SAY ' ';

"EXECIO * DISKR "const.out" (STEM inpl. FINIS)"
IF inpl.Ø=Ø THEN CALL expr(6);

SAY ' ';
SAY 'Searching for Object names';
SAY ' ';

DO i=1 TO inpl.Ø;
   IF pos('_|',inpl.i)<Ø THEN DO;
      obn=obn+1;
      obl.obn=word(inpl.i,2)||'.'||word(inpl.i,4);
   END;
   END;
IF obn=Ø THEN CALL expr(7);

SAY ' ';
SAY '* Tailoring SYSIN DD                                        *';
SAY ' ';

DO i=1 TO obn;
   SELECT;
   WHEN parm.util='RUNSTATS' THEN DO;
      outn=outn+1:outl.outn='RUNSTATS TABLESPACE '||obl.i;
      outn=outn+1:outl.outn='TABLE ALL INDEX(ALL)';
      outn=outn+1:outl.outn='SHRLEVEL CHANGE';
      outn=outn+1:outl.outn='REPORT YES';
      outn=outn+1:outl.outn='UPDATE ALL';
   END;
   WHEN parm.util='REBINDPLAN' THEN DO;
      outn=outn+1:outl.outn='REBIND PLAN('obl.i')';
      outn=outn+1:outl.outn='EXPLAIN(YES)';
   END;
   END;
}
END;
WHEN parm.util='REBINDPACKAGE' THEN DO;
   outn=outn+1;outl.outn='REBIND PACKAGE('obl.i')';
   outn=outn+1;outl.outn='EXPLAIN(YES)';
END;
WHEN parm.util='MODIFYRECOVERYDELETE' THEN DO;
   outn=outn+1;outl.outn='MODIFY RECOVERY TABLESPACE '||obl.i;
   outn=outn+1;outl.outn='DSNUM ALL';
   outn=outn+1;outl.outn='DELETE AGE 185';
END;
OTHERWISE CALL expr(99);
END;

SAY '';
SAY '* Writing tailored SYSIN DD                                 *';
SAY '';

"EXECIO "outn" DISKW "const.jcl" (STEM outl. OPEN FINIS)";
IF rc<>Ø THEN CALL expr(8);

SAY '';
SAY '* Freeing allocated files                                   *';
SAY '';

"FREE FI("const.inp")";
"FREE FI("const.out")";

SAY '*************************************************************';
SAY '* PROCEDURE SUCCESSFULLY ENDED                             *';
SAY '*************************************************************';
SAY '';
EXIT Ø;
expr:
PARSE ARG msgn;
SAY '**** E R R O R **********************************************';
SELECT;
WHEN msgn= 1 THEN SAY 'Allocate FILE('const.inp')   RC='rc;
WHEN msgn= 2 THEN SAY 'Allocate FILE('const.out')   RC='rc;
WHEN msgn= 3 THEN SAY 'Unknown SSID. Allowed: 'const.ssids;
WHEN msgn= 4 THEN SAY 'Program error. Parm.libs value not specified';
WHEN msgn= 5 THEN SAY 'Error executing TSO DSN';
WHEN msgn= 6 THEN SAY 'EXECIO DISKR Empty 'const.out;
WHEN msgn= 7 THEN SAY 'No Objects in File 'const.out;
WHEN msgn= 8 THEN SAY 'EXECIO DISKW Error to 'const.jcl;
WHEN msgn=10 THEN SAY 'Unknown keyword.';
WHEN msgn=11 THEN SAY 'Executing REBIND PLAN or PACKAGE not allowed';
OTHERWISE;
END;
EXIT 16;
RETURN;
IMPORTANT NOTE

Don’t use REBIND PLAN and REBIND PACKAGE administration jobs if you are adding new columns to existing tables. The ‘ALTER TABLE table_name ADD column_name...’ won’t bind a program with the new DCLGEN that incorporates the new columns, particularly if the following statements occur in your program:

- **SELECT** statement that selects all columns, not by listing all column names but by using an asterisk:
  
  ```sql
  EXEC SQL SELECT * INTO ...
  ```

- **INSERT** statement that doesn’t list column names:
  
  ```sql
  EXEC SQL INSERT INTO table_name VALUES ...
  ```

If the REBIND finds one of the above statements, the plan or package will be invalidated. This can only be solved by recompiling and binding with the new DCLGEN that is created after adding the new column.

And what causes this problem? This is, I would say, an IBM inconsistency. If IBM had said that the new columns could be added freely with no effect on existing problems, assuming that the new column is defined as NULL or NOT NULL WITH DEFAULT, they would have had to change how the precompiler worked and created the DBRM.

It is known that if you want to refer to a specific table in your program you have to declare the table you are using. This is usually done by including (EXEC SQL INCLUDE...) the member created by DCLGEN that contains the EXEC SQL DECLARE TABLE statement and the structure that matches the column names and definitions. It is generally used as an interface when sending and receiving data to and from DB2 through SQL statements.

Assume that the precompiler finds the following statement:

```sql
EXEC SQL SELECT * FROM table_name INTO :DCLtable_name
```

where the ‘DCLtable_name’ is the host variable structure created by DCLGEN that describes the table and its columns (number of columns = K). What does the precompiler do? It stores the SQL statement into
DBRM with the structure expanded to its columns, so the statement in the DBRM looks like:

```
EXEC SQL SELECT * FROM table_name INTO :DCLtable_name.column_name1
    :DCLtable_name.column_name2 ... :DCLtable_name.column_nameK.
```

This works OK if the new column is added (new number of columns = K+1) and there are no BINDs and REBINDs. When the program is executed, DB2 replaces the asterisk(*) with the list of column names that existed prior to the new column being added (K columns).

But if you REBIND the program (DBRM), DB2 suddenly fetches the new table definitions and sees that you want to fetch K+1 columns into K host variables and, because of that, reports an error and invalidates the plan or package. The only solution is to create and use new DCLGEN and precompile, compile, and bind the program, because it is a totally new program.

So why didn’t IBM resolve this problem, when it could be simply resolved by using the same policy for expanding table column names as that used for expanding the structure’s columns’ names? The precompiler must be provided with the EXEC SQL DECLARE TABLE statement in your program before the EXEC SQL SELECT statement, so it is easy for the precompiler to fetch the list of column names from that statement and not have to go to the DB2 catalog to fetch them.

Once the column names are retrieved, the precompiler can replace the asterisk with the column names and store it in that way in the DBRM (EXEC SQL SELECT column1, column2...columnK INTO :DCLtable_name.column_name1, :DCLtable_name.column_name2... :DCLtable_name.column_nameK), and then there will be no problem during REBINDs.

The same effect is also seen with the INSERT statement.

So, if IBM could change the way the precompiler works, life could be easier.

---

Josip Ivancic  
Database Administrator  
SDA (Croatia)  

© Xephon 1999
Plan_table Purge Report

INTRODUCTION

The Plan_table Purge Report (PPR) enables the user to clean up the plan table. The utility can work in the following three cases.

Case A

Figure 1 shows a part of the output for the progname XXXXtest. When the last two binds are compared, the access path remains unchanged, leading to duplication.

```
<table>
<thead>
<tr>
<th>Queryno</th>
<th>Qblockno</th>
<th>Planno</th>
<th>Method</th>
<th>Progname</th>
<th>Accesstype</th>
<th>Matchcols</th>
<th>Bind_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-18.59.43.797147</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-18.59.43.797147</td>
</tr>
<tr>
<td>Ø5</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-05-05-15.30.43.097151</td>
</tr>
<tr>
<td>Ø6</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-05-05-15.30.43.097151</td>
</tr>
<tr>
<td>Ø2</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>Ø</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
<tr>
<td>Ø3</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>Ø</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
</tbody>
</table>
```

*Figure 1: Case A*

This would lead to the following report:

**

PROGNAME : XXXXtest
STATUS : NO CHANGE

DELETE RECS FROM PLAN_TB WITH BINDTIME LESS THAN 1998-10-05-18.59.43.797147

**

The result is that all the redundant records for the program are deleted from the plan table.
Case B

Figure 2 shows case B.

When the last two binds are compared, the access path has changed in the Accesstype and matchcols columns.

This would lead to the following report:

```
**
PROGNAME : XXXXtest
STATUS   : CHANGE IN THE ACCESS PATH
CHANGES  : 

QUERYNO : 00000010
QBLOCKNO : 0001
PLANNO : 0001

QUERYNO
BEFORE IMAGE : 0000005
AFTER IMAGE : 0000010

ACCESSTYPE
BEFORE IMAGE : R
AFTER IMAGE : I

MATCHCOLS
BEFORE IMAGE : 0000
AFTER IMAGE : 0002

DELETE RECS FROM PLAN_TB WITH BINDTIME LESS THAN 1998-05-05-15.30.43.097151
```

The result is that records before the specific bind_time are deleted and a report is generated by comparing the last two groups of records for a program.

```
<table>
<thead>
<tr>
<th>Queryno</th>
<th>Oblockno</th>
<th>Planno</th>
<th>Method</th>
<th>Progname</th>
<th>Accesstype</th>
<th>Matchcols</th>
<th>Bind_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-05-05-8.59.43.797147</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>2</td>
<td>1998-05-05-10.59.43.797147</td>
</tr>
<tr>
<td>Ø5</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-05-05-15.30.43.097151</td>
</tr>
<tr>
<td>Ø6</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>R</td>
<td>Ø</td>
<td>1998-05-05-15.30.43.097151</td>
</tr>
<tr>
<td>Ø2</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>R</td>
<td>Ø</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
<tr>
<td>Ø3</td>
<td>1</td>
<td>1</td>
<td>Ø</td>
<td>XXXXtest</td>
<td>R</td>
<td>Ø</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
</tbody>
</table>
```

Figure 2: Case B
Case C

Figure 3 shows case C. The change in the access path is obvious here, with the difference in the number of records for the two versions. This would lead to the following report:

```
**
PROGNAME : XXXXtest
STATUS    : CHANGES FOUND
CHANGES   : NUM OF QUERY STATEMENTS HAS CHANGED

DELETE RECS FROM PLAN_TB WITH BINDTIME LESS THAN 1998-05-01-03.15.43.018141
**

The result is that all the records for the program after a specific bind_time are deleted from the plan table.
```

<table>
<thead>
<tr>
<th>Queryno</th>
<th>Qblockno</th>
<th>Planno</th>
<th>Method</th>
<th>Progname</th>
<th>Accessstype</th>
<th>Matchcols</th>
<th>Bind_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-08.59.43.797147</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-18.59.43.797147</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-18.59.43.797147</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>I</td>
<td>1</td>
<td>1998-10-05-18.59.43.797147</td>
</tr>
<tr>
<td>07</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>R</td>
<td>0</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
<tr>
<td>08</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>XXXXtest</td>
<td>R</td>
<td>0</td>
<td>1998-05-01-03.15.43.018141</td>
</tr>
</tbody>
</table>

**Figure 3: Case C**

USING PPR

To use the PPR utility you should:

- Copy these EXECs into a PDS ‘XXXtest.userid.rexx.explain’ as members EXPLAIN and PURGE.
- Create datasets:
  - XXXtest.userid.JCL.
  - XXXtest.userid.unload.
  - XXXtest.userid.JCL.delete.
  - XXXtest.userid.report.
• Execute the member ‘explain’ and specify the qualifier of the plan_table you want to clean up.

IMPROVEMENTS
Possible improvements could be:
• Have the actual SQL statements in the report. We are currently working on this.
• Enhance performance by having a clustering index on the plan_table.

SELECT
/* ——— REXX TO CLEAN-UP THE PLAN TABLE ———— */
/* — LET THE USER ENTER THE QUALIFIER —————— */

SAY '——————————————————————————————————————————'
SAY 'ENTER THE PLAN_TABLE YOU WANT TO WORK ON :'
SAY '——————————————————————————————————————————'

PULL QUALIFIER

/* — IF THE QUALIFIER IS SPACES ,THEN EXIT ———— */

IF QUALIFIER = ' '
    THEN
        DO
            SAY '———— QUALIFIER IS INVALID —————'
            SAY '- PLEASE DO ENTER A VALID QUALIFIER !! ———'
            EXIT
        END

/* —— THIS JCL UNLOADS THE LAST TWO VERSIONS OF EACH PLAN — */
/* — FROM THE PLAN_TABLE AND EXECUTES THE REXX WHICH —— */
/* — COMPARES AND PREPARES FOR THE DELETE ———— */
/* ————————————————————————————————————————————————————————— */

"ALLOC DS(USERID.INPUT) FI(INP) SHR REU"
NEWSTACK
QUEUE "/XXXXXJOB JOB (ACCT PARAMETER),'COMM..."  "
QUEUE "/ MSGCLASS=X,REGION=ØM,MSGLEVEL=(1,1),NOTIFY=&SYSUID  "
QUEUE "//***********************************************************
QUEUE "/* TO GET THE RECORDS FROM THE PLAN_TABLE
QUEUE "渎************sEKC..."  "
QUEUE "//STEP010 EXEC PGM=IKJEFT01,DYNAMNBR=20,TIME=1439  "

QUEUE "/STEP1 LIB DD DSN=SYS2.DB2.XXXX.DSNLOAD,DISP=SHR"
QUEUE "/SYSTSPRT DD SYSOUT=*"
QUEUE "/SYSPRINT DD SYSOUT=*"
QUEUE "/SYSSOUT DD SYSOUT=*"
QUEUE "/SYSSREC0 DD DSN=XXXTEST.USERID.UNLOAD,DISP=SHR"
QUEUE "/SYSPUNCH DD DUMMY"
QUEUE "/SYSTSIN DD *
QUEUE " DSN SYSTEM(XXX)"
QUEUE " RUN PROGRAM(DSNTIAUL) PARMS('SQL')"
QUEUE " END"
QUEUE "/*
QUEUE "/SYSSIN DD *
QUEUE " SET CURRENT DEGREE = 'ANY';"
QUEUE " SELECT PROGNAME, ' ',BIND_TIME, ' ',DIGITS(QUERYNO) AS "
QUEUE " QUERYNO,DIGITS(QBLOCKNO) AS QBLOCKNO,DIGITS(PLANNO) AS "
QUEUE " PLANNO,DIGITS(METHOD),CREATOR,TNAME,DIGITS(TABNO),"
QUEUE " ACCESSSTYPE,DIGITS(MATCHCOLS),ACCESSCREATOR,"
QUEUE " ACCESSNAME,INDEXONLY,"
QUEUE " SORTN_UNIQ,SORTN_JOIN,SORTN_ORDERBY,SORTN_GROUPBY,"
QUEUE " SORTC_UNIQ,SORTC_JOIN,SORTC_ORDERBY,SORTC_GROUPBY,"
QUEUE " TSLOCKMODE,REMARKS,PREFETCH,COLUMN_FN_EVAL,"
QUEUE " DIGITS(MIXOPSEQ),VERSION,COLLID,DIGITS(ACCESS_DEGREE),"
QUEUE " DIGITS(ACCESS_PGROUP_ID),DIGITS(JOIN_DEGREE),"
QUEUE " DIGITS(JOIN_PGROUP_ID),DIGITS(SORTC_PGROUP_ID),"
QUEUE " DIGITS(SORTN_PGROUP_ID),PARALLELISM_MODE,"
QUEUE " DIGITS(MERGE_JOIN_COLS),CORRELATION_NAME,"
QUEUE " PAGE_RANGE,JOIN_TYPE,GROUP_MEMBER,WHEN_OPTIMIZE,"
QUEUE " QBLOCK_TYPE"
QUEUE " FROM "QUALIFIER".PLAN_TABLE ,"
QUEUE " (SELECT (PROGNAME) AS PGNAME,"
QUEUE " (MAX(BIND_TIME)) AS MAX_BIND"
QUEUE " FROM "QUALIFIER".PLAN_TABLE A"
QUEUE " WHERE BIND_TIME NOT IN"
QUEUE " (SELECT MAX(B.BIND_TIME)"
QUEUE " FROM "QUALIFIER".PLAN_TABLE B"
QUEUE " WHERE A.PROGNAME = B.PROGNAME)"
QUEUE " GROUP BY PROGNAME) AS NEWTAB"
QUEUE " WHERE PROGNAME = PGNAME AND BIND_TIME >= MAX_BIND"
QUEUE " ORDER BY PROGNAME,BIND_TIME DESC,QUERYNO,QBLOCKNO,PLANNO;"
QUEUE "/*
QUEUE "/***********************************************************
QUEUE "//* TO EXEC THE PURGE"
QUEUE "/***********************************************************
QUEUE "/STEP2Ø EXEC PGM=IKJEFTØ1,REGION=4Ø96K,"
QUEUE "/ PARM="PURGE "QUALIFIER""
QUEUE "/SYSEXEC DD DSN=XXXTEST.USERID.REXX.EXPLAIN,DISP=SHR"
QUEUE "/SYSTSPRT DD DSN=XXXTEST.USERID.CHANGES,DISP=SHR"
QUEUE "/SYSTSIN DD DUMMY"
QUEUE "/*
PURGE

/* — REXX TO COMPARE AND PREPARE FOR THE PURGE ——— */
/* — ACCEPTS THE QUALIFIER FROM THE PREVIOUS EXEC ——— */
PARSE ARG QUALIFIER
/* — ALLOCATES THE FILE WHICH HAS THE SORTED ————— */
/* — RECORDS FROM THE PLAN_TABLE ———————————— */
"ALLOC DS(USERID.UNLOAD) FI(INP) SHR REU"
/* — INITIALIZE THE VARIABLES ———————————— */
PLANREC. = ''; I = Ø; FIRST_REC = 'Y'
CALL PRINT_JOB
/* — READS THE FILE INTO A ARRAY ———————————— */
"EXECIO * DISKR INP (STEM PLANREC."
ENDFILE = PLANREC.Ø
/* — SEGREGATE THE TWO VERSIONS INTO DIFFERENT ARRAYS —— */
DO I = I + 1 UNTIL I >= ENDFILE
    IF (FIRST_REC = 'Y')
        THEN
            DO
                CALL ASSIGN
                FIRST_REC = 'N'
            END
        ELSE
            IF (FIRST_REC = 'N')
                THEN
                    DO
                        PG_NM = SUBSTR(PLANREC.I,1,8)
                        BD_TM = SUBSTR(PLANREC.I,12,26)
                        COMP_PG = COMPARE(PG_NM_1,PG_NM)
                        CALL BINDCMP
                        CALL ARRSEG
                    END
                ELSE
                    DO
                        SAY 'INVALID RECORD'
                        EXIT
                    END
            END
    END
CALL DELETE_REC
/* ——— A COMMIT IS ISSUED AT THE END OF THE DELETE ——— */
SAY '———————————————————'
QUEUE " COMMIT;                                                     "
QUEUE "/*
QUEUE ''
"EXECIO * DISKW OUT (FINIS"
"SUBMIT 'XXXTEST.USERID.JCL.DELETE'"
"FREE DD(OUT)"
DELSTACK
EXIT
/* ——— THIS ROUTINE INITIALIZES THE VARIABLES USED —————— */
ASSIGN:
   ARR1. = ' ';ARR2. = ' ';J = 1;K = 1
   PG_NM_1 = SUBSTR(PLANREC.I,1,8)
   BD_TM_1 = SUBSTR(PLANREC.I,12,26)
   BD_TM_2 = '
   ARR1.J  = SUBSTR(PLANREC.I,41)
   J = J + 1
RETURN
DELETE_REC:
/* —— INITIALIZE THE VARIABLES ———————————————— */
   CNT_J = Ø;CNT_K = 1;CHG_FND = 'N'
   SWITCH = 'N';COUNTER = Ø;FIRST_TIME = 'Y'
   DO CNT_J = CNT_J + 1 UNTIL SWITCH = 'Y'
   /* —— COMPARE THE TWO ARRAYS ————————————————— */
   STRINGA = SUBSTR(ARR1.CNT_J,21)
   STRINGB = SUBSTR(ARR2.CNT_K,21)
   COMP_RC = COMPARE(STRINGA,STRINGB)
   /* ——— IF DIFFERENCES ARE FOUND , PERFORM THE REPORT ROUTINE — */
   IF COMP_RC > Ø
      THEN
         DO
            CALL REPORT_CHG
         END
         CNT_K = CNT_K + 1
         IF (CNT_J >= (J - 1) & CNT_K >= (K - 1))
            THEN
               SWITCH = 'Y'
         END
      END
   IF COUNTER = Ø
      THEN
         DO
            SAY ''
            SAY '**'
            SAY 'PROGNAME : 'PG_NM_1
            SAY 'STATUS   : NO CHANGE'
            SAY ''
            SAY 'DELETE RECS FROM PLAN_TB WITH BINDTIME LESS THAN 'BD_TM_1
            BND_TM = BD_TM_1
         END
      ELSE
         DO
            SAY ''
         END
"
SAY 'DELETE RECS FROM PLAN_TB WITH BINDTIME LESS THAN 'BD_TM_2
BND_TM = BD_TM_2
END

/* — PREPARES FOR THE PURGE DEPENDING ON THE BIND_TIME ——— */
/* — FOR EACH PACKAGE ———————————————————— */
QUEUE " DELETE
QUEUE " FROM "QUALIFIER".PLAN_TABLE WHERE
QUEUE " (PROGNAME = '"PG_NM_1"' AND
QUEUE " BND_TIME < '"BND_TM"');
QUEUE " RETURN
REPORT_CHG:
COUNTER = 1
IF J = K
THEN
N0P
ELSE
DO
SAY ''
SAY '***
SAY 'PROGNAME : 'PG_NM_1
SAY 'STATUS   : CHANGES FOUND'
SAY 'CHANGES'
SAY ''
SAY '  NUM OF QUERY STATEMENTS HAVE CHANGED'
SAY ''
COUNTER = 2;SWITCH = 'Y'
RETURN
END
IF FIRST_TIME = 'Y'
THEN
DO
SAY ''
SAY '***
SAY 'PROGNAME : 'PG_NM_1
SAY 'STATUS   : CHANGE IN THE ACCESS PATH'
SAY 'CHANGES'
FIRST_TIME = 'N'
END
SAY ''
SAY 'QUERYNO  : ' SUBSTR(ARR1.CNT_J,1,10)
SAY 'QBLOCKNO : ' SUBSTR(ARR1.CNT_J,11,5)
SAY 'PLANNO   : ' SUBSTR(ARR1.CNT_J,16,5)
SAY ''
DO FOREVER
STRINGA = SUBSTR(ARR1.CNT_J,COUNTER)
STRINGB = SUBSTR(ARR2.CNT_K,COUNTER)
COMP_RC = COMPARE(STRINGA,STRINGB)
IF COMP_RC > Ø
THEN
DO
  COUNTER = (COUNTER - 1) + COMP_RC
  CALL MAP
END
ELSE
  LEAVE
END RETURN
/* —— THIS ROUTINE MAPS THE DIFFERENCES TO THE COLUMNS OF ——— */
/* —— THE PLAN TABLE ————————————————————— */
MAP:
SELECT
  WHEN (COUNTER >= 1 & COUNTER <= 10) THEN
    DO
      SAY '          QUERYNO'
      SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,1,10)
      SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,1,10)
      COUNTER = 11
    END
  WHEN (COUNTER >= 11 & COUNTER <= 15) THEN
    DO
      SAY '          QBLOCKNO'
      SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,11,5)
      SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,11,5)
      COUNTER = 16
    END
  WHEN (COUNTER >= 16 & COUNTER <= 20) THEN
    DO
      SAY '          PLANNO'
      SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,16,5)
      SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,16,5)
      COUNTER = 21
    END
  WHEN (COUNTER >= 21 & COUNTER <= 25) THEN
    DO
      SAY '          METHOD'
      SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,21,5)
      SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,21,5)
      COUNTER = 26
    END
  WHEN (COUNTER >= 26 & COUNTER <= 33) THEN
    DO
      SAY '          CREATOR'
      SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,26,8)
      SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,26,8)
      COUNTER = 34
    END
  WHEN (COUNTER >= 34 & COUNTER <= 51) THEN
    DO
      SAY '          TNAME'
      SAY '          BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,34,51)
      SAY '          AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,34,51)
      COUNTER = 52
    END
END
SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,34,18)
SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,34,18)
COUNTER = 52
END
WHEN (COUNTER >= 52 & COUNTER <= 56) THEN
  DO
    SAY '            TABNO'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,52,5)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,52,5)
    COUNTER = 57
  END
WHEN (COUNTER >= 57 & COUNTER <= 58) THEN
  DO
    SAY '            ACCESSTYPE'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,57,2)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,57,2)
    COUNTER = 59
  END
WHEN (COUNTER >= 59 & COUNTER <= 63) THEN
  DO
    SAY '            MATCHCOLS'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,59,5)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,59,5)
    COUNTER = 64
  END
WHEN (COUNTER >= 64 & COUNTER <= 71) THEN
  DO
    SAY '            ACCESSCREATOR'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,64,8)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,64,8)
    COUNTER = 72
  END
WHEN (COUNTER >= 72 & COUNTER <= 89) THEN
  DO
    SAY '            ACCESSNAME'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,72,18)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,72,18)
    COUNTER = 90
  END
WHEN (COUNTER = 90) THEN
  DO
    SAY '            INDEXONLY'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,90,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,90,1)
    COUNTER = 91
  END
WHEN (COUNTER = 91) THEN
  DO
    SAY '            SORTN_UNIQ'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,91,1)
SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,91,1)
COUNTER = 92
END

WHEN (COUNTER = 92) THEN
  DO
    SAY '          SORTN_JOIN'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,92,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,92,1)
    COUNTER = 93
  END

WHEN (COUNTER = 93) THEN
  DO
    SAY '          SORTN_ORDERBY'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,93,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,93,1)
    COUNTER = 94
  END

WHEN (COUNTER = 94) THEN
  DO
    SAY '          SORTN_GROUPBY'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,94,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,94,1)
    COUNTER = 95
  END

WHEN (COUNTER = 95) THEN
  DO
    SAY '          SORTC_UNIQ'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,95,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,95,1)
    COUNTER = 96
  END

WHEN (COUNTER = 96) THEN
  DO
    SAY '          SORTC_JOIN'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,96,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,96,1)
    COUNTER = 97
  END

WHEN (COUNTER = 97) THEN
  DO
    SAY '          SORTC_ORDERBY'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,97,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,97,1)
    COUNTER = 98
  END

WHEN (COUNTER = 98) THEN
  DO
    SAY '          SORTC_GROUPBY'
    SAY '            BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,98,1)
    SAY '            AFTER  IMAGE :' SUBSTR(ARR1.CNT_J,98,1)
COUNTER = 99
END

WHEN (COUNTER >= 99 & COUNTER <= 101) THEN
DO
  SAY 'TSLOCKMODE'
  SAY 'BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,99,3)
  SAY 'AFTER IMAGE :' SUBSTR(ARR1.CNT_J,99,3)
  COUNTER = 102
END

WHEN (COUNTER >= 102 & COUNTER <= 357) THEN
DO
  SAY 'REMARKS'
  SAY 'COLUMN HAS BEEN CHANGED'
  COUNTER = 358
END

WHEN (COUNTER = 358) THEN
DO
  SAY 'PREFETCH'
  SAY 'BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,358,1)
  SAY 'AFTER IMAGE :' SUBSTR(ARR1.CNT_J,358,1)
  COUNTER = 359
END

WHEN (COUNTER = 359) THEN
DO
  SAY 'COLUMN_FN_ENVAL'
  SAY 'BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,359,1)
  SAY 'AFTER IMAGE :' SUBSTR(ARR1.CNT_J,359,1)
  COUNTER = 360
END

WHEN (COUNTER >= 360 & COUNTER <= 364) THEN
DO
  SAY 'MIXOPSEQ'
  SAY 'BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,360,5)
  SAY 'AFTER IMAGE :' SUBSTR(ARR1.CNT_J,360,5)
  COUNTER = 365
END

WHEN (COUNTER >= 365 & COUNTER <= 430) THEN
DO
  SAY 'VERSION'
  SAY 'VERSION COLUMN HAS CHANGED'
  SAY 'PL BROWSE THE TABLE FOR CONTENTS'
  COUNTER = 431
END

WHEN (COUNTER >= 431 & COUNTER <= 448) THEN
DO
  SAY 'COLLID'
  SAY 'BEFORE IMAGE :' SUBSTR(ARR2.CNT_K,431,18)
  SAY 'AFTER IMAGE :' SUBSTR(ARR1.CNT_J,431,18)
  COUNTER = 449
END
WHEN (COUNTER >= 449 & COUNTER <= 454) THEN
  DO
    SAY '          ACCESS_DEGREE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,449,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,449,5)
  COUNTER = 455
END
WHEN (COUNTER >= 455 & COUNTER <= 460) THEN
  DO
    SAY '          ACCESS_PGROUP_ID'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,455,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,455,5)
  COUNTER = 461
END
WHEN (COUNTER >= 461 & COUNTER <= 466) THEN
  DO
    SAY '          JOIN_DEGREE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,461,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,461,5)
  COUNTER = 467
END
WHEN (COUNTER >= 467 & COUNTER <= 472) THEN
  DO
    SAY '          JOIN_PGROUP_ID'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,467,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,467,5)
  COUNTER = 473
END
WHEN (COUNTER >= 473 & COUNTER <= 478) THEN
  DO
    SAY '          SORTC_PGROUP_ID'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,473,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,473,5)
  COUNTER = 479
END
WHEN (COUNTER >= 479 & COUNTER <= 484) THEN
  DO
    SAY '          SORTN_PGROUP_ID'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,479,5)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,479,5)
  COUNTER = 485
END
WHEN (COUNTER >= 485 & COUNTER <= 486) THEN
  DO
    SAY '          PARALLELISM_MODE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,485,1)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,485,1)
  COUNTER = 487
END
WHEN (COUNTER >= 487 & COUNTER <= 492) THEN
DO
  SAY '          MERGE_JOIN_COLS'
  SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,487,5)
  SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,487,5)
  COUNTER = 493
END

WHEN (COUNTER >= 493 & COUNTER <= 511) THEN
  DO
    SAY '          CORRELATION_NAME'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,493,18)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,493,18)
    COUNTER = 512
  END

WHEN (COUNTER = 512) THEN
  DO
    SAY '          PAGE_RANGE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,512,1)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,512,1)
    COUNTER = 513
  END

WHEN (COUNTER = 513) THEN
  DO
    SAY '          JOIN_TYPE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,513,1)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,513,1)
    COUNTER = 514
  END

WHEN (COUNTER >= 514 & COUNTER <= 521) THEN
  DO
    SAY '          GROUP_MEMBER'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,514,8)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,514,8)
    COUNTER = 522
  END

WHEN (COUNTER = 522) THEN
  DO
    SAY '          WHEN_OPTIMIZE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,522,1)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,522,1)
    COUNTER = 523
  END

WHEN (COUNTER >= 523 & COUNTER <= 528) THEN
  DO
    SAY '          QBLOCK_TYPE'
    SAY '            BEFORE IMAGE : ' SUBSTR(ARR2.CNT_K,523,6)
    SAY '            AFTER  IMAGE : ' SUBSTR(ARR1.CNT_J,523,6)
    COUNTER = 529
  END

OTHERWISE
  DO
    
  END

/* —— QUEUES THE JCL WHICH PURGES THE RECORDS FOR THE PLAN —— */
/* —— DEPENDING ON THE BIND_TIME ———————————————— */
PRINT_JOB:
"ALLOC DS(USERID.JCL.DELETE) FI(OUT) SHR"
NEWSTACK
QUEUE "/XXXXXJOB JOB (ACCT PARAMETER),'COMMENTS',CLASS=A,"
QUEUE "/ MSGCLASS=X,REGION=ØM,MSGLEVEL=(1,1),NOTIFY=&SYSUID"
QUEUE "********************************************************************************
QUEUE "/ DELETE THE RECORDS"
QUEUE "********************************************************************************
QUEUE "/STEPØ1 EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(4,LT)"
QUEUE "/STEPØ1B DD DSN=SYS2.DB2.XXXX.DSNLOAD,DISP=SHR"
QUEUE "/SYSSTSPRT DD SYSOUT=*"
QUEUE "/SYSPRINT DD SYSOUT=*"
QUEUE "/SYSOUT DD SYSOUT=*"
QUEUE "/SYSRECV0 DD SYSOUT=*"
QUEUE "/SYSMPUNCH DD DUMMY"
QUEUE "/SYSSTSN DD *
QUEUE " DSN SYSTEM(XXXX)"
QUEUE " RUN PROGRAM(DSNTIAUL) PARMS('SQL')"
QUEUE " END"
QUEUE "*/"
QUEUE "/SYSIN DD *
QUEUE " SET CURRENT DEGREE = 'ANY' ;"
RETURN
/* —— THESE SUBROUTINES ARE TO HAVE THE VERSIONS IN DIFFERENT — */
/* —— ARRAYS —————————————————————————— */
BINDCMP:
   IF (K > 1 & COMP_PG = Ø)
      THEN
      DO
         COMP_BD = COMPARE(BD_TM_2,BD_TM)
      END
   ELSE
   IF (K = 1 & COMP_PG = Ø)
      THEN
      DO
         COMP_BD = COMPARE(BD_TM_1,BD_TM)
      END
   ELSE
   IF (COMP_PG > Ø)
      THEN
      DO
         COMP_BD = COMPARE(BD_TM_1,BD_TM_2)
      END
ELSE
  DO
    SAY 'PROBLEM IN PRONAME'
    EXIT
  END
RETURN
ARRSEG:
  IF (COMP_PG = Ø & COMP_BD = Ø)
  THEN
    DO
      IF K > 1
        THEN
          DO
            ARR2.K  = SUBSTR(PLANREC.I,41)
            K = K + 1
          END
        ELSE
          DO
            ARR1.J = SUBSTR(PLANREC.I,41)
            J = J + 1
          END
        END
    ELSE
      IF (COMP_PG = Ø & COMP_BD > Ø)
        THEN
          DO
            BD_TM_2 = SUBSTR(PLANREC.I,12,26)
            ARR2.K  = SUBSTR(PLANREC.I,41)
            K = K + 1
          END
        ELSE
          IF (COMP_PG > Ø & COMP_BD > Ø)
            THEN
              DO
                CALL DELETE_REC
                CALL ASSIGN
              END
            ELSE
              DO
                SAY 'BIND_TIME PROBLEM'
                EXIT
              END
            END
        END
    ELSE
      DO
        SAY 'PROBLEM IN PRONAME'
        EXIT
      END
  END
RETURN

Prasad V Taiwade, K R Swaminaathan, and Kiran Haryadi
Wipro Infotech (India) © Wipro Infotech 1999
IBM has announced an upgrade to DB2 for OS/390 Version 5, with the addition of the DB2 Management Tools Package and DB2 Control Centre. The latter provides a graphical interface for managing the database on OS/390, Unix, Windows, and OS/2 systems, making it easier to obtain, apply, and install the current service along with the program. New functions and enhancements target scalability, availability, accessibility, and management.

The upgrade includes improved performance for certain SQL queries, like joins with CHAR data types of unequal length, list prefetch with index screening, and uncorrelated subquery with an indexable predicate. Also improved is DRDA access for applications that can use DESCRIBE INPUT parameter and greater SQL compatibility with support for VARCHAR columns of up to 255 characters.

For further information contact your local IBM representative.

* * *

DB2 users can benefit from Version 2.1 of BMC’s InTune OS/390 application performance tuner, enabling the sharing of information between systems via parallel sysplex, to analyse the target application and simplify customization of batch reports.

Version 2.1 identifies application program delays and presents this information for analysis through an interactive interface for both traditional and parallel sysplex environments. In-built support for parallel sysplex allows organizations to share performance information between various systems within the sysplex, allowing users to choose all or any specific individual systems when invoking requests.

For further information contact:
BMC Software, 2101 CityWest Boulevard, Houston, TX 77042-2827, USA.
Tel: (713) 918 8800.
BMC Software, Compass House, 207-215 London Road, Camberley, Surrey, GU15 3EY, UK.
Tel: (01276) 24622.

* * *

UBS AG has announced CARES for DB2, for keeping DB2 subsystems in top condition. CARES conducts many of the daily routine checks necessary for preventative maintenance. It identifies, reports, and helps to eliminate potential inhibitors in the area of CPU and I/O performance, utility and application concurrency, space savings, parallel sysplex, etc.

For further information contact:
UBS AG, Keytools, Hochstrasse 16, Postfach CH-4002 Basel, Switzerland.
Tel: (61) 288 3057.
Relational Architects International, 33 Newark Street, Hoboken, NJ 07030, USA.
Tel: (201) 420 0400.

* * *