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; you will need the user-id shown on your address label.

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 £245.00 in the UK; $365.00 in the USA and Canada; £251.00 in Europe; £257.00 in Australasia and Japan; and £255.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £21.00 ($31.00) each including postage.
Running 24-bit CAF applications

Starting with DB2 Version 4.10, some changes have been implemented in the CAF language interface addressing mode – for example DSNALI AMODE changed from any to 31. The changes to DSNALI module attributes AMODE and RMODE are shown in Figure 1.

<table>
<thead>
<tr>
<th>DB2 VERSION</th>
<th>3.10</th>
<th>4.10</th>
<th>5.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMODE</td>
<td>24</td>
<td>ANY</td>
<td>ANY</td>
</tr>
<tr>
<td>AMODE</td>
<td>ANY</td>
<td>31</td>
<td>31</td>
</tr>
</tbody>
</table>

*Figure 1: DSNALI module attributes*

For old COBOL/VS programs, after migration to DB2 Version 4.10 or 5.10, you may experience 0C4 abends in programs with AMODE=24 using CAF – if these programs don’t use a proper addressing mode switching instruction (BASSM).

We had this problem with COBOL/VS modules used in an ‘old’ commercial software package. Because it was a software package, it was not possible for us to recompile or modify these modules.

A BYPASS SOLUTION

To solve this problem we had to write a ‘transparent’ DSNALI module as an interface between the application programs and DSNALI.

This module doesn’t do anything except load ‘real’ IBM DSNALI and call it using a BASSM instruction.

The DSNALI module has several entry points to handle dynamic calls. They are:

- ‘* DSNALI’ – for DB2 connections.
• ‘* DSNHLI2’ – for SQL calls.
• ‘* DSNWLI2’ – for IFI calls.

Our ‘transparent’ modules are called with ‘real’ IBM names (DSNALI, DSNHLI2, and DSNWLI2).

‘Real’ IBM modules were ‘copied’ to DSNALI3, DSNHLI3, and DSNWLI3. Figure 2 describes interconnections between the different modules involved in that process.

Figure 2: Interconnections between modules

INSTALLATION PROCEDURE

Firstly, we used SMP/E to copy real IBM modules and to rename the new modules (DSNXXX3). In this way, if a PTF modifies one of the CAF modules, DSNXXX3 modules are also modified. This guarantees that our DB2 environment is kept valid.

```c
//*
//SMP EXEC PGM=GIMSMP,REGION=4M,
PARM='DATE=U,CSI=SMAINT.DB2.V510.CSI'
//*
//SMPHOLD DD DUMMY
//SMPCNTL DD *
SET BDY (GLOBAL).
REJECT SELECT(DB20001) BYPASS(APPLYCHECK) .
RESETRC .
```
RECEIVE SELECT(DB20001).
SET BDY (DSNTARG).
APPLY SELECT(DB20001)
   REDO .
/*
/*
/*
//SMPPTFIN DD DATA,DLM='$$'
++USERMOD(DB20001).
++VER(P115) FMINID(HDB5510).
++MOD(DSNALI) LKLIB(ADSNLOAD).
++MOD(DSNAA) LKLIB(ADSNLOAD).
++JCLIN .
//LINKØ1 EXEC PGM=HEWL,PARM='RENT,AMODE=31,RMODE=ANY,NCAL'
//SYSPRT N DD SYSOUT=* 
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DISP=SHR,DSN=DB2.SDSNLOAD
//SYSLIN DD *
   INCLUDE ADSNLOAD(DSNALI)
   INCLUDE ADSNLOAD(DSNAA)
   ORDER DSNAA
   ENTRY DSNALI3(R)
/*
//LINKØ2 EXEC PGM=HEWL,PARM='RENT,AMODE=31,RMODE=ANY,NCAL'
//SYSPRT N DD SYSOUT=* 
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DISP=SHR,DSN=DB2.SDSNLOAD
//SYSLIN DD *
   INCLUDE ADSNLOAD(DSNALI)
   INCLUDE ADSNLOAD(DSNAA)
   ORDER DSNAA
   ENTRY DSNHLI3(R)
/*
//LINKØ3 EXEC PGM=HEWL,PARM='RENT,AMODE=31,RMODE=ANY,NCAL'
//SYSPRT N DD SYSOUT=* 
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DISP=SHR,DSN=DB2.SDSNLOAD
//SYSLIN DD *
   INCLUDE ADSNLOAD(DSNALI)
   INCLUDE ADSNLOAD(DSNAA)
   ORDER DSNAA
   ENTRY DSNHLI3(R)
/*
$$
/*
Secondly, we had to write and compile our ‘transparent’ modules. The source code of our three modules uses exactly the same logic:
• It loads the ‘real’ IBM module once.
• It then passes control to the IBM module without doing anything
  – it can’t even modify register values!

The code for the ‘transparent’ modules follows.

DSNALI

```assembly
DSNALI CSECT
DSNALI AMODE 24
DSNALI RMODE 24
*
  SAVE (14,12)
  BASR R12,Ø
  USING *,R12         R12 = BASE REGISTER
*
  GETMAIN R,LV=WORKL
*
  ST   R1,8(R13)
  ST   R13,4(R1)
  LR   R13,R1
  USING WORK,R13
*
  ST   R12,REG12
*
  CLC  LISQL,=F'Ø'    FIRST CALL ?
  BNE  CALLIT        NO, JUST CALL IT
*
  MVC  LOAD(LOADL),LOADM   LOAD IBM MODULE
  LOAD  EP=DSNALI3,SF=(E,LOAD)
  ST   RØ,LISQL
*
CALLIT EQU *
*
  L   R14,SAVEAREA+4
*
  L   R14,12(R14)
  L   R15,LISQL
  LM   RØ,R12,2Ø(R14)
*
  BASSM R14,R15
*
  L   R12,REG12
*
RETURN L   R13,4(R13)     RESTORE R13
  L   R1,8(R13)
  FREEMAIN R,LV=WORKL,A=(R1)
  L   R14,12(R13)
```

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
LM  R0,R12,20(R13)
SR  R15,R15
BR  R14

* LOADM  LOAD  EP=DSNALI3,SF=L
LOADL  EQU  *-LOADM
*
LISQL  DC  F'0'
*
WORK  DSECT
SAVEAREA  DS  18F
SAVE  DS  18F
REG12  DS  F
LOAD  DS  CL(LOADL)
WORKL  EQU  *-WORK
*
   REGISTER
*
END

DSNHIL2

CCDSNHIL2  CSECT
DSNHIL2  AMODE 24
DSNHIL2  RMODE 24
*
   SAVE  (14,12)
   BASR  R12,0
   USING  *,R12  R12 = BASE REGISTER
*
   GETMAIN  R,lv=WORKL
*
   ST  R1,8(R13)
   ST  R13,4(R1)
   LR  R13,R1
   USING  WORK,R13
*
   ST  R12,REG12
*
   CLC  LISQL,F'0'  FIRST CALL?
   BNE  CALLIT  NO, JUST CALL IT
*
   MVC  LOAD(LOADL),LOADM  LOAD IBM MODULE
   LOAD  EP=DSNHIL3,SF=(E,LOAD)
   ST  R0,LISQL
*
   CALLIT  EQU  *
*
   L  R14,SAVEAREA+4
* L R14,12(R14)
  L R15, LISQL
  LM RØ, R12, 2Ø(R14)
  *
  BASM R14, R15
  *
  L R12, REG12
  *
  RETURN L R13, 4(R13) RESTORE R13
  L R1, 8(R13)
  FREEMAIN R, LV=WORKL, A=(R1)
  L R14, 12(R13)
  LM RØ, R12, 2Ø(R13)
  SR R15, R15 SET UP RC
  BR R14 RETURN TO MVS AND USE RC=R15
  *
  LOADM LOAD EP=DSNHLI3, SF=L
  LOADL EQU *-LOADM
  *
  LISQL DC F'Ø'
  *
  WORK DSECT
  SAVEAREA DS 18F
  SAVE DS 18F
  REG12 DS F
  LOAD DS CL(LOADL)
  WORKL EQU *-WORK
  *
  REGISTER
  *
  END

DSNWLI2

DSNWLI2 CSECT
DSNWLI2 AMODE 24
DSNWLI2 RMODE 24
*
  SAVE (14, 12)
  BASR R12, Ø
  USING *, R12 R12 = BASE REGISTER
  *
  GETMAIN R, LV=WORKL
  *
  ST R1, 8(R13)
  ST R13, 4(R1)
  LR R13, R1
  USING WORK, R13
  *
  ST R12, REG12
*  CLC LISQL,-F'Ø' FIRST CALL ?
 BNE CALLIT NO, JUST CALL IT
*  MVC LOAD(LOADL),LOADM LOAD IBM MODULE
 LOAD EP=DSNWLI3, SF=(E,LOAD)
 ST RØ,LISQL
*
 CALLIT EQU *
*
 L R14,SAVEAREA+4
 L R14,12(R14)
 L R15,LISQL
 LM RØ,R12,2Ø(R14)
*
 BASSM R14,R15
*
 L R12,REG12
*
 RETURN L R13,4(R13) RESTORE R13
 L R1,8(R13)
 FREEMAIN R, LV=WORKL, A=(R1)
 L R14,12(R13)
 LM RØ,R12,2Ø(R13)
 SR R15,R15 SET UP RC
 BR R14 RETURN TO MVS AND USE RC=R15
*
 LOADM LOAD EP=DSNHLI3, SF=L
 LOADL EQU *-LOADM
*
 LISQL DC F'Ø'
*
 WORK DSECT
 SAVEAREA DS 18F
 SAVE DS 18F
 REG12 DS F
 LOAD DS CL(LOADL)
 WORKL EQU *-WORK
*
 REGISTER
*
 END

LINK-EDIT INSTRUCTIONS
To work properly, these ‘transparent’ modules must be link-edited as re-usable.

Patrick Renard
CTRNE (France) © Xephon 1998
A tool for checking space status

There may be an occasion in the life of a DBA when simply missing one small thing can lead to big trouble at some later time. A DBA might forget a routine task that needs be performed regularly at a specific time, even if the task is considered important and is written in the database housekeeping manual.

For example, a DBA may be responsible for ensuring that the production database can be used smoothly by on-line users throughout the day. This objective could not be met if some spaces in the databases are restricted in status or some DB2 utilities are running.

The tablespace or indexspace are restricted if:

- It is started for read-only processing, utility-only processing, or stopped.
- It is waiting for deferred start.
- It is being processed by a utility.
- It is in copy pending, check pending, or recovery pending status.
- It contains a page-error range.

The DBA has to run DB2 utilities regularly to maintain a DB2 system at peak performance. One of the most frequently used DB2 utilities is COPY for back-up. At most sites, before image copy is invoked, all spaces are set to read-only status to avoid changes to the database, and after the back-up has finished they are set to read/write again.

Sometimes the job doesn’t run well, and it is possible that the operator may forget to inform the DBA. The result is that, the next morning, users complain to the Help Desk, which later contacts the DBA.

A possible solution is to provide a tool that informs the DBA when something has happened in the database. By doing so, the DBA can prepare for the worst-case scenario at the earliest possible time, in fact at the time that he/she first logs on to TSO that day!

This program is a CLIST program called CHKREST, which checks the output of ‘DISPLAY DATABASE’ and ‘DISPLAY UTILITY’
commands in all the DB2 regions. If there are restricted spaces or running utilities, it will notify the responsible DBAs or Production Control staff.

By using this, the DBA can be confident that life will be easier – because any problem can be fixed before the users start to complain or are even aware of the problem!

The CLIST is called from JCL that should run regularly (by OPC or other scheduler in MVS) every morning before the on-line users log-on. If the users log-on at 7:00 am, then it can be set up to run at 6:30 am every morning. The message is sent to the DBA using the TSO command XMIT (TRANSMIT), so the DBA should use the TSO command RECEIVE at every log-on to TSO. This means you have to add a RECEIVE line at the beginning of a CLIST program (LOGPROF) that is invoked by the log-on procedure each time you log-on. The CLIST program is normally found in ICQ.ICQCCLLIB(LOGPROF).

The RECEIVE command could be inserted as the first command as shown below.

**LOGPROF CLIST**

```clist
PROC Ø
SET &DSNAME = &SYSUID..ISPF.ISPPROF /* SET DEFAULT NAME */
CONTROL NOMSG NOFLUSH
/****************************************************************************
/ * CHANGE : */
/ * BY : TGDH+TGIPTN2+TGIPTN1 */
/ * REQ : RUDIARF */
/ * DES : EXECUTE TSO COMMAND RECEIVE TO GET DB2 MESSAGE */
/****************************************************************************
RECEIVE
FREE F((ISPPROF)
ALLOC DA('&DSNAME') F(ISPPROF) OLD

**CHKREST JCL**

```
/ * STEPØ1Ø - WRITE OUTPUT OF DSN COMMAND -DIS DB/-DIS UTIL 
/ * THERE ARE 3 DB2 SUBSYSTEMS TO BE CHECKED IN THIS 
/ * SAMPLE : DB2T, DSNS, AND DSNP 
/ * STEPØ2Ø - RUN CLIST TO CHECK THE OUTPUT OF STEPØ1Ø AND NOTIFY 
/ * TO DBAS IF NEEDED 
/ * CLIST PARM : 1. OUTPUT OF STEPØ1Ø (SYSTSPRT) 
/ * 2. DBA'S TSO-ID TO BE NOTIFIED 
/ * YOU COULD ADD MORE TSO IDS AS PARM, BUT THE CLIST 
/ * PROGRAM SHOULD BE MODIFIED 
/ * NOTE : DB2, TSO/E, JES2 AND OPC/ESA ARE REQUIRED 
/ ******************************************
/STEPØ1Ø EXEC PGM=IKJEFTØ1 
/STEPLIB DD DSN=DSN31ØCD.SDSNLOAD,DISP=SHR
/SYSTSPRT DD DSN=RUDIARF.DSNOUT.DATA,DISP=SHR
/SYSDUMP DD SYSSOUT=* 
/SYSTSIN DD * 
DSN  SYSTEM(DB2T)
  -DIS UTIL(*)
  -DIS DB(*) SPACE(*) RES
DSN  SYSTEM(DSNS)
  -DIS UTIL(*)
  -DIS DB(*) SPACE(*) RES
DSN  SYSTEM(DSNP)
  -DIS UTIL(*)
  -DIS DB(*) SPACE(*) RES
END
/STEPØ2Ø EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø 
/SYSTSPRT DD SYSSOUT=* 
/SYSPRINT DD SYSSOUT=* 
/SYSTSIN DD * 
  EXEC 'RUDIARF.CLIST(CHKREST)' 'RUDIARF.DSNOUT.DATA RUDIARF'
/*

CHKREST CLIST
PROC 2 DSIN NOTIFY1 
CONTROL NOSYMLIST NOCONLIST

/ ******************************************/
/*. CHKREST - CHECK RESTRICTED SPACES AND RUNNING UTILITIES */
/*. IN DATABASES WHICH REGULARLY MONITORED BY DBA */
/*. */
/*. STEPS : */
/*. 1.READ OUTPUT OF DSN COMMAND (-DISPLAY DB/ -DISPLAY UTIL) */
/*. 2.IF THERE ARE RESTRICTED SPACE IN DB, WRITE THE DB */
/*. 3.IF THERE ARE RUNNING UTILITIES IN DB, WRITE THE DB */
/*. 4.NOTIFY THE MESSAGE FILE TO THE DBAS IN CHARGE */
/*. */
/*. SCHEDULING INFORMATION: */
/*. FREQUENCY : DAILY */
/*. */
/*. */
/* PARAMETERS */
/* - DSIN : THE OUTPUT OF DSN COMMAND -DIS DB/-DIS UTIL */
/* - NOTIFY1 : THE TSO-ID OF DBAS IN CHARGE */
/* YOU CAN ADD MORE DBA TSO-ID AS PARAMETERS BY ADDING NOTIFY2. */
/* AFTER NOTIFY1 AT THIS PROGRAM'S FIRST LINE, AND ADD 2 LINES. */
/* 'XMIT <JES2 SYSTEM-ID>.NOTIFY2 ...' BELOW AT THE BOTTOM */
/* */
/* */
/* INPUTS (STEP, DESCRIPTION) */
/* 1. FILEIN: THE OUTPUT OF DSN COMMAND -DIS DB/-DIS UTIL */
/* */
/* OUTPUT (STEP, DESCRIPTION) */
/* 4. FILEMSG: MESSAGE FILE TO BE SENT */
/* */
/* UPDATE LOG: */
/* DD/MM/YY -UPD BY—— DECRPIPTION OF CHANGE—— */
/* 01/15/98 RUDI ARIEF CREATE INITIAL VERSION */

SET EOF=OFF
SET DB_OK=FALSE
SET UTIL_OK=FALSE
SET IS_DB=FALSE
SET IS_UTIL=FALSE
SET SSID=NULL
SET RG_PAR=&STR())
SET SSID_LINE=&STR(DSN  SYSTEM)
SET DB_LINE=&STR(  -DIS DB)
SET UTIL_LINE=&STR(  -DIS UTIL)
SET ERR_DB_SSID=NULL
SET ERR_UTIL_SSID=NULL

/*** CHECK END-OF-FILE ***/
ERROR +
DO
  /* EOF CAUSES ERROR CODE 400 */
  IF &LASTCC=400 THEN +
    DO
      SET EOF=ON
    END
  RETURN
END

/*** READ THE OUTPUT OF THE -DISPLAY UTIL/DB ***/
ALLOCATE FILE(FILEIN) DA('&DSIN') SHR REU
OPENFILE FILEIN
DO WHILE 1=1
  GETFILE FILEIN
  IF &EOF=ON THEN GOTO OUT

  /* CHECK IF THERE IS MSG ' DSN SYSTEM(...)' */
  SET LOC = &SYSINDEX(&STR(&SSID_LINE),&STR(&FILEIN))
  IF &LOC > Ø THEN +

DO
SET LOC2 = &SYSINDEX(&STR(&RG_PAR),&STR(&FILEIN))-1
SET LOC = &LOC+12
SET SSID=&SUBSTR(&LOC:&LOC2,&FILEIN)
END

/* CHECK IF THE COMMAND IS DISPLAY UTILITY */
SET LOC = &SYSINDEX(&STR(&UTIL_LINE),&STR(&FILEIN))
IF &LOC > Ø THEN +
  DO
    SET IS_UTIL=TRUE
  END

/* CHECK IF THE COMMAND IS DISPLAY DATABASE */
SET LOC = &SYSINDEX(&STR(&DB_LINE),&STR(&FILEIN))
IF &LOC > Ø THEN +
  DO
    SET IS_DB=TRUE
  END

/* CHECK IF THERE IS MSG 'DSNT365I  NO DATABASES FOUND' */
SET LOC = &SYSINDEX(DSNT365I,&FILEIN)
IF &LOC > Ø THEN +
  DO
    SET DB_OK=TRUE
  END

/* CHECK IF THERE IS MSG 'DSNU112I  NO AUTHORIZED UTILITY FOUND' */
SET LOC = &SYSINDEX(DSNU112I,&FILEIN)
IF &LOC > Ø THEN +
  DO
    SET UTIL_OK=TRUE
  END

/* CHECK IF THERE IS MSG 'DSN9022I  DISPLAY DATABASE COMPLETED' */
SET LOC = &SYSINDEX(DSN9022I,&FILEIN)
IF &LOC > Ø THEN +
  DO
    /* CHECK IF THE DATABASES ARE OK */
    IF &IS_DB=TRUE AND &DB_OK=FALSE THEN +
      DO
        IF &ERR_DB_SSID=NULL THEN +
          SET ERR_DB_SSID = &SSID
        ELSE +
          SET ERR_DB_SSID = &ERR_DB_SSID &SSID
        END
      END
      /* CHECK IF THE UTILITIES ARE OK */
      IF &IS_UTIL=TRUE AND &UTIL_OK=FALSE THEN +
        DO
          IF &ERR_UTIL_SSID=NULL THEN +
            SET ERR_UTIL_SSID = &SSID
          END
        END
  END
END
ELSE +
    SET ERR_UTIL_SSID = &ERR_UTIL_SSID &SSID
END

/* RESET FLAGS */
SET DB_OK = FALSE
SET UTIL_OK = FALSE
SET IS_DB = FALSE
SET IS_UTIL = FALSE
END
END

/* WRITE THE MESSAGE FILE AND SEND TO THE DBAS */
OUT: CLOSFILE FILEIN

IF &ERR_UTIL_SSID¨=NULL OR +
&ERR_DB_SSID¨=NULL THEN +
DO
/* WRITE TO MESSAGE FILE */
ALLOCATE FILE(FILEMSG) DA('RUDIARF.MSG') SHR REU
OPENFILE FILEMSG OUTPUT
SET &FILEMSG=.............. WARNING ..............
PUTFILE FILEMSG
IF &ERR_UTIL_SSID¨=NULL THEN +
DO
    SET &FILEMSG = RUNNING UTILITIES IN DATABASE: &ERR_UTIL_SSID
    PUTFILE FILEMSG
END
IF &ERR_DB_SSID¨=NULL THEN +
DO
    SET &FILEMSG = RESTRICTED SPACES IN DATABASE: &ERR_DB_SSID
    PUTFILE FILEMSG
END
CLOSFILE FILEMSG
/* NOTIFY THE ERROR MESSAGE TO THE DBAS IN CHARGE */
/* THE DESTINATION IS YOUR JES2 SITE NAME + DBA'S TSO ID */
XMIT MVS1JES2.&NOTIFY1 MSGDATASET('RUDIARF.MSG') NOLOG NONOTIFY
XMIT MVS1JES2.&NOTIFY1 MSGDATASET('&DSIN') NOLOG NONOTIFY
END
EXIT

As an alternative, you could run the DB2 commands ‘DISPLAY DATABASE’ and ‘DISPLAY UTILITY’ directly each time you log-on (by adding it to LOGPROF CLIST). However, this is not the preferred approach because it will make your log-on time slightly longer (while you wait for the command to be processed).

Rudi Arief
DBA
Caltex Pacific (Indonesia)
Rebind and convert plans and packages – part 3

This month we complete the code that enables you to rebind plans and packages or convert plans to packages.

REBP01

)Attr Default(%+_ | type(text) intens(high) caps(on) color(white)  
    hilite(reverse) # type(text) intens(high) caps(off) color(green)  
    \ type(output) intens(high) caps(off) color(yellow)  
    [ type(input) intens(high) just(left) pad('_')  
)body window(62,17)  
    | Rebind Plans  
    +  
      + SSID[db2 + Runstat:+[rst+ +Keep temporary file:+[ans+  
      + ==>[ff+`msg  
      + -------------------------------  
      # 1-Rebind plan(s)[ppla +  
      # 2-Rebind all plans  
      # 3-Rebind all plans bound before a given date and time  
      # 4-Rebind all plans bound since a given date and time  
      # 5-Rebind all plans bound within a given date and time range  
      # 6-Rebind all invalid plans  
      # 7-Rebind all inoperative plans  
      # 8-Rebind all plans bound with isolation level CS  
      + -------------------------------  
      + Bdate1[date1 +Btime1[time1 +Bdate2[date2 +Btime2[time2 +  
      + -------------------------------  
      | PF3 End +  
)init  
    IF (&ff = 1,2,3,4,5,6,7,8,1?,2?,3?,4?,5?,6?,7?,8?)  
      &msg = ''  
    ELSE  
      &msg = 'Enter 1, .... to 8 or 1?, .... to 8? for help'  
)proc  
    VPUT (db2,ppla,date1,date2,time1,time2) PROFILE  
)end

REBP02

)Attr Default(%+_ | type(text) intens(high) caps(on) color(white)  
    hilite(reverse) # type(text) intens(high) caps(off) color(green)  
    \ type(output) intens(high) caps(off) color(yellow)  
    [ type(input) intens(high) just(left) pad('_')  
    ""
Rebind Packages

1. Rebind package(s)
2. Rebind all versions of the packages
3. Rebind all packages bound before a given date and time
4. Rebind all packages bound since a given date and time
5. Rebind all packages bound within a given date and time range
6. Rebind all invalid versions of the packages
7. Rebind all inoperative versions of the packages
8. Rebind all packages that allow CPU and/or I/O parallelism

Tst1[tst1] Tst2[tst2]

% msg = ''
ELSE
(msg = 'Enter 1, .... to 8 or 1?, .... to 8? for help')

VPUT (db2,pack,tst1,tst2) PROFILE

REBP01H

Attr Default(%+_) | type(text) intens(high) caps(on ) color(white)
hilite(reverse) / type(text) intens(high) caps(off ) color(yellow) \ 
type(output) intens(high) caps(off ) color(white) hilite(reverse)
% type(output) intens(high) caps(off ) color(green)
[ type(output) intens(high) caps(off ) color(red)
_ type(input) color(red) hilite(uscore) intens(high)
Conversion plan to package

+ SSID[db2 + Runstat:+[rst+
\msg +
+ -----------------------------------------------------------
# Enter plan name(s) to be convert:
+ ] Plan name:+[cplan +
+ # Change current defaults?:+[def+
+ -----------------------------------------------------------
| Enter to Continue + | PF3 Return +
)init
)proc
  VPUT (db2,cplan) PROFILE
)end

Conversion plan to package

+ SSID[db2 + Runstat:+[rst+
\msg +
+ -----------------------------------------------------------
# Enter plan name(s) to be converted - DB2 wildcards supported
+ @ Plan name:+[cplan +
+ # Change current defaults?:+[def+
+ -----------------------------------------------------------
# Isolation :+[iso + #Validate:+]val +
# Release :+[rel + #Explain :+]exp +
# Currentdata :+]cda + #Degree :+]deg +
# Dynamicrules:+j dru +
| Enter to Continue + | PF3 Return +
init
proc
  VPUT (iso,val,rel,exp,cda,deg,dru) PROFILE
end

RSQLPAN
)Attr Default(%+_) ( type(text ) intens(high) hilite(reverse)
  ] type(text ) intens(high) hilite(reverse) color(green)
  / type(text ) intens(high) hilite(reverse) color(yellow)
  ¬ type( input ) intens(high) color(red)
  % type(text ) intens(high)
  + type(text ) intens(low )
  _ type( input ) intens(high) caps(on ) just(left )
  ¬ type( input ) intens(low ) caps(off) just(asis )
)Body window(76,19)
(Result SQL query+
  +Command ===>._zcmd +Scroll
===>_amt +
)Model
¬z
+
)Init
  .ZVARS = '(v1)'
  &amt = PAGE
)Reinit
)Proc
)End

PREB00
PREBØØ1 .ALARM = YES .WINDOW=NORESP .ALARM = YES'&message'

PREBPL
* PROCESS GS,OFFSET,OPT(TIME); PREBPL:PROC(PARMS)OPTIONS(MAIN) REORDER:
/***********************************************************************
/* DESCRIPTION: REBIND PLANS                                              */
/************************************************************************
DCL PARMS CHAR(100) VAR;
DCL SYSPRINT FILE STREAM OUTPUT;
DCL NUMSEQ BIN FIXED(31) INIT(Ø);
DCL 1 WORKST.
EXEC SQL INCLUDE SQLCA;

/* SELECTION 1 */
IF SUBSTR(PARMS,1,1)='1' THEN DO;
  IF SUBSTR(PARMS,2,8)=' ' THEN PLAN='%';
  ELSE DO;
    CALL FUNC(SUBSTR(PARMS,2,8),OUT);
    PLAN=OUT;
    IF LENGTH(PLAN) < 8 THEN PLAN=PLAN||'%';
  END;
END;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BCREATOR, BNAME
FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
WHERE NAME LIKE :PLAN
  AND NAME = DNAME
  AND BTYPE='R'
FOR FETCH ONLY;

/* SELECTION 2 */
IF SUBSTR(PARMS,1,1)='2' THEN DO;
  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
  SELECT DISTINCT BCREATOR, BNAME
  FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
  WHERE NAME = DNAME
  AND BTYPE='R'
  FOR FETCH ONLY;
END;

/* SELECTION 3 */
IF SUBSTR(PARMS,1,1)='3' THEN DO;
  BDATE1 = SUBSTR(PARMS,10,6);
  BTIME1 = SUBSTR(PARMS,16,8);
  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
  SELECT DISTINCT BCREATOR, BNAME
  FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
  WHERE NAME = DNAME
  AND BINDDATE <= :BDATE1
  AND BINDTIME <= :BTIME1
  AND BTYPE='R'
  FOR FETCH ONLY;
END;
END;
COMMENT SELECTION 4 */
IF SUBSTR(PARMS,1,1)='4' THEN DO;
    BDATE1 = SUBSTR(PARMS,10,6);
    BTIME1 = SUBSTR(PARMS,16,8);
    EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
    SELECT DISTINCT BCREATOR, BNAME
    FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
    WHERE NAME = DNAME
        AND BINDDATE >= :BDATE1
        AND BINDTIME >= :BTIME1
        AND BTYPE='R'
    FOR FETCH ONLY;
END;
COMMENT SELECTION 5 */
IF SUBSTR(PARMS,1,1)='5' THEN DO;
    BDATE1 = SUBSTR(PARMS,10,6);
    BTIME1 = SUBSTR(PARMS,16,8);
    BDATE2 = SUBSTR(PARMS,24,6);
    BTIME2 = SUBSTR(PARMS,30,8);
    EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
    SELECT DISTINCT BCREATOR, BNAME
    FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
    WHERE NAME = DNAME
        AND BINDDATE >= :BDATE1 AND BINDTIME >= :BTIME1
        AND BINDDATE <= :BDATE2 AND BINDTIME <= :BTIME2
        AND BTYPE='R'
    FOR FETCH ONLY;
END;
COMMENT SELECTION 6 */
IF SUBSTR(PARMS,1,1)='6' THEN DO;
    EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
    SELECT DISTINCT BCREATOR, BNAME
    FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
    WHERE VALID='N'
        AND BTYPE='R'
    FOR FETCH ONLY;
END;
COMMENT SELECTION 7 */
IF SUBSTR(PARMS,1,1)='7' THEN DO;
    EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
    SELECT DISTINCT BCREATOR, BNAME
    FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
    WHERE OPERATIVE='N'
        AND BTYPE='R'
    FOR FETCH ONLY;
END;
COMMENT SELECTION 8 */
IF SUBSTR(PARMS,1,1)='8' THEN DO;
    EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR

SELECT DISTINCT BCREATOR, BNAME
FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
WHERE ISOLATION='S'
    AND BTYPE='R'
FOR FETCH ONLY;
END;

EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
DO WHILE (SQLCODE=Ø);
    NUMSEQ=1;
    PUT SKIP LIST (SUBSTR(BCREATOR,1,8)||BNAME);
    EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');

FUNC:PROC(INP,OUT);
    DCL IC  BIN FIXED(15);
    DCL INP CHAR(8);
    DCL OUT CHAR(8) VAR;
    DO IC=1 TO 8 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
    END;
    OUT= SUBSTR(INP,1,IC-1);
END FUNC;
END PREBPL;

PREBPA
* PROCESS GS,OFFSET,OPT(TIME); PREBPA:PROC(PARMS)OPTIONS(MAIN) REORDER:
/***************************************************************************/
/* DESCRIPTION: REBIND PACKAGES */
/***************************************************************************/
DCL PARMS CHAR(100) VAR;
DCL SYSPRINT  FILE STREAM OUTPUT;
DCL NUMSEQ BIN FIXED(31) INIT(Ø);
DCL 1 WORKST,
    2 SEL      CHAR(1) ,
    2 PACK     CHAR(8) VAR,
    2 TSTAMP1  CHAR(26) ,
    2 TSTAMP2  CHAR(26) ,
    2 BCREATOR CHAR(8) VAR,
    2 BNAME    CHAR(18) VAR;
DCL OUT     CHAR(8) VAR;
DCL (SUBSTR,NULL,ADDR,LENGTH) BUILTIN;

EXEC SQL INCLUDE SQLCA;

/* SELECTION 1 */
IF SUBSTR(PARMS,1,1)='1' THEN DO;
   IF SUBSTR(PARMS,2,8)=' ' THEN PACK='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,2,8),OUT);
   PACK=OUT;
   IF LENGTH(PACK) < 8 THEN PACK||'%;
END;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
   ,SYSIBM.SYSPACKDEP
   WHERE LOCATION = DLOCATION
   AND COLLID = DCOLLID
   AND NAME = DNAME
   AND CONTOKEN = DCONTOKEN
   AND NAME LIKE :PACK
   AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 2                                   */
IF SUBSTR(PARMS,1,1)='2' THEN DO;
   EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
   ,SYSIBM.SYSPACKDEP
   WHERE LOCATION = DLOCATION
   AND COLLID = DCOLLID
   AND NAME = DNAME
   AND CONTOKEN = DCONTOKEN
   AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 3                                   */
IF SUBSTR(PARMS,1,1)='3' THEN DO;
   TSTAMP1 = SUBSTR(PARMS,10,26);
   EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
   ,SYSIBM.SYSPACKDEP
   WHERE LOCATION = DLOCATION
   AND COLLID = DCOLLID
   AND NAME = DNAME
   AND CONTOKEN = DCONTOKEN
   AND BINDTIME <= :TSTAMP1
   AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 4                                   */
IF SUBSTR(PARMS,1,1)='4' THEN DO;
   TSTAMP1 = SUBSTR(PARMS,10,26);
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND BINDTIME <= :TSTAMP1
  AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 5 */
IF SUBSTR(PARMS,1,1)='5' THEN DO;
  TSTAMP1 = SUBSTR(PARMS,1,26);
  TSTAMP2 = SUBSTR(PARMS,26,26);
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND BINDTIME >= :TSTAMP1
  AND BINDTIME <= :TSTAMP2
  AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 6 */
IF SUBSTR(PARMS,1,1)='6' THEN DO;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND VALID='N'
  AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 7 */
IF SUBSTR(PARMS,1,1)='7' THEN DO;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND BINDTIME <= :TSTAMP1
  AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 8 */
IF SUBSTR(PARMS,1,1)='8' THEN DO;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND BINDTIME >= :TSTAMP1
  AND BINDTIME <= :TSTAMP2
  AND BTYPE='R'
FOR FETCH ONLY;
END;
/* SELECTION 9 */
IF SUBSTR(PARMS,1,1)='9' THEN DO;
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BQUALIFIER, BNAME
FROM SYSIBM.SYSPACKAGE
  ,SYSIBM.SYSPACKDEP
WHERE LOCATION = DLOCATION
  AND COLLID   = DCOLLID
  AND NAME     = DNAME
  AND CONTOKEN = DCONTOKEN
  AND VALID='N'
  AND BTYPE='R'
FOR FETCH ONLY;
END;
AND COLLID  = DCOLLID
AND NAME     = DNAME
AND CONTOKEN = DCONTOKEN
AND OPERATIVE='N'
AND BTYPE='R'
FOR FETCH ONLY;
END;

/* SELECTION 8                         */
IF SUBSTR(PARMS,1,1)='8' THEN DO;
   EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
   SELECT DISTINCT BQUALIFIER, BNAME
   FROM SYSIBM.SYSPACKAGE
 ,SYSIBM.SYSPACKDEP
   WHERE LOCATION = DLOCATION
   AND COLLID  = DCOLLID
   AND NAME     = DNAME
   AND CONTOKEN = DCONTOKEN
   AND DEGREE='ANY'
   AND BTYPE='R'
   FOR FETCH ONLY;
END;
EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
DO WHILE (SQLCODE=0);
   NUMSEQ=1;
   PUT SKIP LIST (SUBSTR(BCREATOR,1,8)||BNAME);
   EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');

FUNC:PROC(INP,OUT);
   DCL IC  BIN FIXED(15);
   DCL INP CHAR(8);
   DCL OUT CHAR(8) VAR;
   DO IC=1 TO 8 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
   END;
   OUT=SUBSTR(INP,1,IC-1);
END FUNC;
END PREBPA;

PREBCO

* PROCESS GS,OFFSET,OPT(TIME); PREBCO:PROC(PARMS)OPTIONS(MAIN) REORDER;                        */
/*******************************************************************************/
DESCRIPTION: CONVERSION PLAN TO PACKAGE                       */
*******************************************************************************/
DCL PARMS CHAR(100) VAR;
DCL SYSPRINT FILE STREAM OUTPUT;
DCL NUMSEQ BIN FIXED(31) INIT(Ø);
DCL 1 WORKST,
  2 PLAN CHAR(8) VAR,
  2 NEW CHAR(1) .
  2 ISO CHAR(4) .
  2 VAL CHAR(4) .
  2 REL CHAR(10) .
  2 EXP CHAR(4) .
  2 CDA CHAR(4) .
  2 DEG CHAR(4) .
  2 DRU CHAR(4) :
DCL 1 WORKVA,
  2 ISOP CHAR(2) VAR,
  2 VALP CHAR(4) VAR,
  2 RELP CHAR(10) VAR,
  2 EXPP CHAR(3) VAR,
  2 CDAP CHAR(3) VAR,
  2 DEGP CHAR(3) VAR,
  2 DRUP CHAR(4) VAR;
DCL BCREATOR CHAR(8) VAR;
DCL BNAME CHAR(18) VAR;
DCL CSIZE PIC'ZZZZ9';
DCL OUT CHAR(44) VAR;
DCL (SUBSTR,NULL,ADDR,LENGTH) BUILTIN;
EXEC SQL INCLUDE SQLCA;

/****************************************************************/
/* DCLGEN TABLE: SYSIBM.SYSDBRM                                 */
/****************************************************************/
DCL 1 DCLD,
  5 DNAME CHAR(8),
  5 PDSNAME CHAR(44) VAR,
  5 PLNAME CHAR(8),
  5 PLCREATOR CHAR(8);

/****************************************************************/
/* DCLGEN TABLE: SYSIBM.SYSPLAN                                 */
/****************************************************************/
DCL 1 DCLP,
  5 NAME CHAR(8),
  5 CREATOR CHAR(8),
  5 VALIDATE CHAR(1),
  5 ISOLATION CHAR(1),
  5 RELEASE CHAR(1),
  5 EXPLAN CHAR(1),
  5 EXPREDICATE CHAR(1),
  5 QUALIFIER CHAR(8),
  5 CACHESIZE BIN FIXED(15),
/* GET INPUT PARAMETERS                          */
PLAN=SUBSTR(PARMS,1,8);
CALL FUNC(PLAN,OUT);
PLAN=OUT;
ISO=SUBSTR(PARMS,10,4);
VAL=SUBSTR(PARMS,14,4);
REL=SUBSTR(PARMS,18,10);
EXP=SUBSTR(PARMS,28,4);
CDA=SUBSTR(PARMS,32,4);
DEG=SUBSTR(PARMS,36,4);
DRU=SUBSTR(PARMS,40,4);
CALL FUNC(SUBSTR(PARMS,1,8),OUT);
PLAN=OUT;
IF LENGTH(PLAN) < 8 THEN PLAN=PLAN||'%';
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT NAME,PDSNAME,PLNAME,PLCREATOR
FROM SYSIBM.SYSDBRM
WHERE PLNAME LIKE :PLAN
FOR FETCH ONLY;
EXEC SQL OPEN C1;
EXEC SQL FETCH C1 INTO :DNAME, :PDSNAME, :PLNAME, :PLCREATOR;
DO WHILE (SQLCODE=0):;
EXEC SQL SELECT
   NAME, CREATOR, VALIDATE, ISOLATION,
   RELEASE, EXPLAN, EXPREDICATE, QUALIFIER
   CACHESIZE, DEGREE, DYNAMICRULES
INTO
   :NAME, :CREATOR, :VALIDATE, :ISOLATION,
   :RELEASE, :EXPLAN, :EXPREDICATE, :QUALIFIER,
   :CACHESIZE, :DEGREE, :DYNAMICRULES
FROM SYSIBM.SYSPLAN
WHERE NAME=PLNAME
WITH CS;
EXEC SQL FETCH C1 INTO :DNAME, :PDSNAME, :PLNAME, :PLCREATOR;
END;
EXEC SQL CLOSE C1;
PACK:PROC;
CSIZE=CACHESIZE;
/* THE CURRENTDATA OPTION                          */
IF CDA = 'SAME'
THEN DO;
   IF EXPREDICATE = 'C' THEN CDAP='YES';
   IF EXPREDICATE = 'B' THEN CDAP='NO';
ELSE CDAP=CDA;
/* THE DEGREE OPTION */
IF DEG = 'SAME'
THEN DO;
    IF DEGREE = 'ANY' THEN DEGP='ANY';
    IF DEGREE='1' | DEGREE=' ' THEN DEGP='1';
END;
ELSE DEGP=DEG;
/* THE DYNAMICRULES OPTION */
IF DRU = 'SAME'
THEN DO;
    IF DYNAMICRULES='B' THEN DRUP='BIND';
    IF DYNAMICRULES=' ' THEN DRUP='RUN';
END;
ELSE DRUP=DRU;
/* EXPLAIN OPTION FOR THE PACKAGE */
IF EXP = 'SAME'
THEN DO;
    IF EXPLAN='Y' THEN EXPP='YES';
    IF EXPLAN='N' THEN EXPP='NO';
END;
ELSE EXPP=EXP;
/* ISOLATION LEVEL FOR THE PACKAGE */
IF ISO = 'SAME'
THEN DO;
    IF ISOLATION='R' THEN ISOP='RR';
    IF ISOLATION='S' THEN ISOP='CS';
    IF ISOLATION='U' THEN ISOP='UR';
END;
ELSE ISOP=ISO;
/* RELEASE OPTION FOR THE PACKAGE */
IF REL = 'SAME'
THEN DO;
    IF RELEASE='C' THEN RELP='COMMIT';
    IF RELEASE='D' THEN RELP='DEALLOCATE';
END;
ELSE RELP=REL;
/* VALIDATE OPTION FOR THE PACKAGE */
IF VAL = 'SAME'
THEN DO;
    IF VALIDATE='B' THEN VALP='BIND';
    IF VALIDATE='R' THEN VALP='RUN';
END;
ELSE VALP=VAL;
call func(plname,OUT);
call func(dname,OUT);
call func(pdsname,OUT);

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
PUT SKIP LIST ('    LIBRARY('''||OUT||''') -');
PUT SKIP LIST ('    ACTION(REPLACE) -');
CALL FUNC(PLCREATOR,OUT);
PUT SKIP LIST ('    OWNER('||OUT||') -');
PUT SKIP LIST ('    QUALIFIER('||OUT||') -');
CALL FUNC(CDAP,OUT);
PUT SKIP LIST ('    CURRENTDATA('||OUT||') -');
CALL FUNC(DEGP,OUT);
PUT SKIP LIST ('    DEGREE('||OUT||') -');
CALL FUNC(DRUP,OUT);
PUT SKIP LIST ('    DYNAMICRULES('||OUT||') -');
CALL FUNC(EXPP,OUT);
PUT SKIP LIST ('    EXPLAIN('||OUT||') -');
CALL FUNC(ISOP,OUT);
PUT SKIP LIST ('    ISOLATION('||OUT||') -');
CALL FUNC(RELP,OUT);
PUT SKIP LIST ('    RELEASE('||OUT||') -');
CALL FUNC(VALP,OUT);
PUT SKIP LIST ('    VALIDATE('||OUT||')');
CALL FUNC(NAME,OUT);
PUT SKIP LIST ('BIND PLAN('||OUT||') -');
CALL FUNC(CREATOR,OUT);
PUT SKIP LIST ('    PKLIST('||OUT||'.*) -');
CALL FUNC(REPLACE,OUT);
PUT SKIP LIST ('    OWNER('||OUT||') -');
PUT SKIP LIST ('    QUALIFIER('||OUT||') -');
PUT SKIP LIST ('    CACHESIZE('||CSIZE||') -');
PUT SKIP LIST ('    ISOLATION(CS) -');
PUT SKIP LIST ('    VALIDATE(BIND)');
END PACK;
FUNC:PROC(INP,OUT);
DCL IC  BIN FIXED(15);
DCL INP CHAR(44);
DCL OUT CHAR(44) VAR;
DO IC=1 TO 44 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
END;
OUT=SUBSTR(INP,1,IC-1);
END FUNC;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
END PREBCO;

PREBRU

* PROCESS GS,OFFSET,OPT(TIME); PREBRU:PROC(PARMS)OPTIONS(MAIN) REORDER;
/%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%*/
/* DESCRIPTION: DISTINCT DATABASE TABLESPACE FOR RUNSTAT */
/%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%*/
DCL PARMS CHAR(100) VAR;
DCL SYSPRINT FILE STREAM OUTPUT;

DCL NUMSEQ    BIN FIXED(31) INIT(Ø);
DCL 1 WORKST,
    2 PLAN    CHAR(8) VAR,
    2 BCREATOR  CHAR(8) VAR,
    2 BNAME    CHAR(18) VAR;
DCL OUT       CHAR(8) VAR;
DCL (SUBSTR,NULL,ADDR,LENGTH) BUILTIN;

EXEC SQL INCLUDE SQLCA;

/* GET INPUT PLAN(S) */
CALL FUNC(SUBSTR(PARMS,1,8),OUT);
PLAN=OUT;
IF LENGTH(PLAN) < 8 THEN PLAN=PLAN||'%';
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT DISTINCT BCREATOR, BNAME
FROM SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP
WHERE NAME LIKE :PLAN
    AND NAME = DNAME
    AND BTYPE='R'
FOR FETCH ONLY;

EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
DO WHILE (SQLCODE=Ø);
    NUMSEQ=1;
    PUT SKIP LIST (SUBSTR(BCREATOR,1,8)||BNAME);
    EXEC SQL FETCH C1 INTO :BCREATOR, :BNAME;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');

FUNC:PROC(INP,OUT);
    DCL IC  BIN FIXED(15);
    DCL INP CHAR(8);
    DCL OUT CHAR(8) VAR;
    DO IC=1 TO 8 BY 1 WHILE (SUBSTR(INP,IC,1) != ' ');
    END;
    OUT=SUBSTR(INP,1,IC-1);
END FUNC;
END PREBRU;

PLANREB
)

CM Skeleton to generate JCL for rebind plan(s)

----------

//&user.X JOB (ACCT#),"&option",
// NOTIFY=&user,REGION=4M,
// CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//** *******************************************************************
//** &title
//** Date:&dat
//** Time:&tim
//** *******************************************************************
//
//)SEL &rst = YES
//**---- TERMINATE UTILITY ---------------------
//TERMUTIL EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
// DSN SYSTEM(&db2)
// TERM UTILITY(&user..RUNSTA)
//
//**---- RUNSTATS ------------------------------
//RUN30 EXEC DSNUPROC,SYSTEM=&db2,COND=(4,LT),
// UID='&user..RUNSTA',UTPROC='' 
//STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR 
//SYSSIN DD *
//)DOT "PLPA"
//)BLANK 1
// RUNSTATS TABLESPACE &dbname..&tsname
// TABLE (ALL)
// INDEX (ALL)
// SHRLEVEL REFERENCE
// REPORT NO
// UPDATE ALL
//)ENDDOT
//
//)ENDSEL
//)DELOLD EXEC PGM=IDCAMS,COND=(4,LT)
//SYSPRINT DD SYSOUT=* 
//SYSSIN DD *
// DELETE '&user..SYSTSIN.DATA'
// SET MAXCC = Ø 
//
//)UNLOAD EXEC PGM=IKJEFT01
//STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR 
//SYSTSPRT DD SYSOUT=* 
//SYSSIN DD *
// DSN SYSTEM(&db2)
// RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB41) PARM('SQL') -
// LIB('DSN410.RUNLIB.LOAD')
//)END
//SYSPRINT DD SYSOUT=* 
//SYSPUNCH DD SYSOUT=* 
//SYSPUNCH DD SYSOUT=*
/* GENERATE SUBCOMMANDS TO REBIND PLANS OR PACKAGES */

/* STRIP THE BLANKS OUT OF THE REBIND SUBCOMMANDS AND PUT IN THE DSN COMMAND STATEMENTS */

/* EXECUTE THE REBIND SUBCOMMANDS THROUGH DSN */
CONVERSE

)TBA 72
)CM -------------------------------------------------------------
)CM Skeleton to generate JCL for conversion plan to package -------
)CM -------------------------------------------------------------
/&user.X JOB (ACCT#),'&option'.
  NOTIFY=&user,REGION=4M,
  CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
/** ****************************************************************
  **
  ** Date:&dat
  ** Time:&tim
  ** ****************************************************************
/**
)SEL &rst = YES
/**----- TERMINATE UTILITY -----------------------------
/TERMUTIL EXEC PGM=IKJEFT01,COND=(4,LT)
/STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR
/SYSTSPRT DD SYSOUT=* 
/SYSTSIN DD * 
  DSN SYSTEM(&db2)
    -TERM UTILITY(&user..RUNSTA)
/**
//----- RUNSTATS -----------------------------
/RUN3Ø EXEC DSNUPROC,SYSTEM=&db2,COND=(4,LT).
  UID='&user..RUNSTA',UTPROC=''
/STEPLIB DD DSN=DSN410.SDSNLOAD,DISP=SHR
/SYSIN DD *
)DOT "PLPA"
Capturing DISPLAY BUFFERPOOL output

In this article we present a method for capturing the output from the DISPLAY BUFFERPOOL command into a set of DB2 tables for manipulation with SQL. This was considered to be useful for monitoring trends in an application or workload during specific time intervals. Before explaining the methodology, a slight detour will be taken to run through some of the ideas that led to this technique for investigating the buffer pool.

BACKGROUND

Prior to the availability of the DISPLAY BUFFERPOOL command, the information available to investigate the buffer pool was the
statistical information from the SMF type100 records, or the IFCID 198 to obtain real-time trace information.

The SMF record interval is driven from the DSNZPARM parameter, and is normally set at 15 minutes.

This interval duration is not sufficiently short to provide analysis at the level of granularity required for buffer pool investigations – although this data is a very good source for long term analysis and as a management reporting tool.

The IFCID 198 is the best source for investigating the optimum buffer pool size required for a particular level of performance. However, the problems associated with using this trace in a commercial environment deter its use because:

- There is a high overhead of CPU usage, and the output produced can be overwhelming.
- To obtain the best results, a dedicated environment has to be set up for such an analysis. The costs associated with this cannot be ignored.

The advantage of using this technique, if feasible, is the quality and detail of information available. In Figure 1, you can see a portion of the output produced during a test run. Each record consists of the ACE, sequence number, timestamp, buffer pool-id, database-id, pageset-id, page number, type of GETPAGE, buffer hit/miss, and type of access. These fields are more fully explained in the macro DSNDQW02 provided with DB2 software.

Although the information is available, its analysis requires great care and thought.

Much insight and help in this direction was obtained from DB2 buffer pools and page sets: Matching their characteristics for better performance, by Chuck Hoover of Compuware Corporation. The complexity and the overhead in obtaining the data means that this technique is not feasible in a commercial environment without deep pockets.

This was the spur to investigate the output of the DISPLAY
BUFFERPOOL command and add to the repertoire of possible methods for investigating the buffer pool.

**METHOD**

The output of the DISPLAY BUFFERPOOL with the LSTATS option was analysed for a snapshot look at the buffer pool. The full command was:

```
DISPLAY BUFFERPOOL(ACTIVE) DETAIL(INTERVAL) LSTATS.
```

The purpose was to set up a DB2 table structure such that the output of this command would be captured and loaded into this structure. At present the program listed below only gathers together pageset and virtual buffer pool information.

The program will soon be modified and the table structures extended to capture the hiper pool information.

The table definitions are shown later. The table structure was designed to follow the message numbers identified with the various outputs of the DISPLAY BUFFERPOOL command.
A typical load statement is shown below – the load of the other tables is similar. The source code reads the output produced by the DISPLAY BUFFERPOOL command and produces a flat file which is loaded into the tables. To ensure this is loaded into the correct tables, the output records are appended with a two-byte identifier as shown below:

TØ4DSNB42XI ‘Ø6’
TØ5DSNB43XI ‘Ø7’
TØ6DSNB41XI ‘Ø5’
TØ7DSNB4ØXI ‘Ø1’
TØ8DSNB45XI ‘Ø2’
TØ9DSNB455I ‘Ø3’
T1ØDSNB456I ‘Ø4’

The timestamp information is collected rather crudely, but is sufficient for the purposes of recording the time of execution of the DISPLAY BUFFERPOOL command.

A sample JCL is provided for executing the program and generating the flat file to load into the respective tables.

An example of virtual buffer pool ratios used to monitor the effectiveness of the storage usage is provided. We will not go into the details of these ratios since the information has already been explained in Point-in-time DB2 buffer pool reporting, DB2 Update, Issue 54, April 1997.

CONCLUSION

The real problem with reactive analysis is that it can be too reactive, or that it falls into the same category as the old ways, where the buffer pool was simply increased until one obtained diminishing returns in terms of hit ratios and/or the system begins to suffer paging problems. The method used is a slight variation which provides something slightly more durable in that the data can be captured at small intervals for a short duration – say, at 30 second intervals over a duration of five minutes.

This will provide slices of buffer pool and pageset data over a period – but slices that are very close to each other. Also, loading the data into DB2 tables provides opportunities for comparing a similar duration
over several days. Obtaining such data provides the granularity discussed above and also provides some idea of trend analysis.

One of the added benefits of the DISPLAY BUFFERPOOL command is that it provides information at the pageset level. The information is restricted to changed pages, cached pages, I/O delay times, and number of I/Os (both synchronous and asynchronous) of the open pagesets at the time of the execution of the command. It is still useful enough to build a profile of a pageset’s occupancy of the buffer pool during the interval.

It does not, of course, provide the GETPAGE information given by the IFCID 198 trace. Therefore, analysis of the same type cannot be done on the output of the DISPLAY BUFFERPOOL command.

It would be challenging to attempt to model this data to see if any useful conclusions can be reached to quantify buffer pool sizing.

The next phase is to design table structures and provide a program to capture the global buffer pool information via the common DISPLAY GROUPBUFFER POOL.

The advent of DB2 Version 4.1, data sharing, and the coupling facility, makes identification of problems in this area of paramount importance.

TABLE DEFINITIONS

CREATE TABLE T04DSNB42XI
(XXXXBPNM CHAR(4) NOT NULL, bufferpool-id
 XXXXSTME CHAR(15) NOT NULL, stored time
 XXXXRTEM CHAR(25) NOT NULL, reported time
 XXXXSPGU INTEGER NOT NULL WITH DEFAULT, system pages updated
 XXXXSPGW INTEGER NOT NULL WITH DEFAULT, system pages written
 XXXXASWI INTEGER NOT NULL WITH DEFAULT, asynch write I/Os
 XXXXSYWI INTEGER NOT NULL WITH DEFAULT, synch write I/Os
 XXXXDWTH INTEGER NOT NULL WITH DEFAULT, deferred write thresholds
 XXXXVWWT INTEGER NOT NULL WITH DEFAULT, vertical DWTH
 XXXXNWEW INTEGER NOT NULL WITH DEFAULT) no write engines
IN XXXX05D XXXX81S;

CREATE TABLE T05DSNB43XI
(XXXXBPNM CHAR(4) NOT NULL,
 XXXXSTME CHAR(15) NOT NULL,
 XXXXRTEM CHAR(25) NOT NULL,
 XXXXNSYR INTEGER NOT NULL WITH DEFAULT, sync HP->VP without ADMF
 XXXXNSYW INTEGER NOT NULL WITH DEFAULT, sync VP->HP without ADMF

IN XXXXØ5D XXXX81S    ;
CREATE TABLE TØ6DSNB41XI
(XXXXBPNM CHAR(4) NOT NULL,
 XXXXSTME CHAR(15) NOT NULL,
 XXXXRTME CHAR(25) NOT NULL,
 XXXXRGP Pag INTEGER NOT NULL WITH DEFAULT, random getpage
 XXXXSR1O INTEGER NOT NULL WITH DEFAULT, sync I/Os for RGPG
 XXXXSSG Pag INTEGER NOT NULL WITH DEFAULT, sequential getpqge
 XXXXSSSI INTEGER NOT NULL WITH DEFAULT, sync I/Os for SGPG
 XXXXSDMTH INTEGER NOT NULL WITH DEFAULT, data manager threshold
 XXXXSPRQ INTEGER NOT NULL WITH DEFAULT, sequential prefetch requests
 XXXXSSPIO INTEGER NOT NULL WITH DEFAULT, sequential prefetch I/Os
 XXXXSPG R INTEGER NOT NULL WITH DEFAULT, pages read by SPIO
 XXXXXLP RQ INTEGER NOT NULL WITH DEFAULT, list prefetch requests
 XXXXXLP IO INTEGER NOT NULL WITH DEFAULT, list prefetch I/Os
 XXXXXLPGr INTEGER NOT NULL WITH DEFAULT, pages read by LPIO
 XXXXXDP RQ INTEGER NOT NULL WITH DEFAULT, dynamic prefetch requests
 XXXXXDP IO INTEGER NOT NULL WITH DEFAULT, dynamic prefetch I/Os
 XXXXXDPGr INTEGER NOT NULL WITH DEFAULT, pages read by DPIO
 XXXXXPNOB INTEGER NOT NULL WITH DEFAULT, prefetch disabled no buffers
 XXXXXPNRE INTEGER NOT NULL WITH DEFAULT) prefetch disabled no read engines
IN XXXXØ5D XXXX81S    ;
CREATE TABLE TØ7DSNB4ØXI
(XXXXBPNM CHAR(4) NOT NULL,
 XXXXSTME CHAR(15) NOT NULL,
 XXXXBPUC SMALLINT NOT NULL WITH DEFAULT, VP use count
 XXXXBPsz INTEGER NOT NULL WITH DEFAULT, VP size
 XXXXBPAL INTEGER NOT NULL WITH DEFAULT, VP buffers allocated
 XXXXBPDE INTEGER NOT NULL WITH DEFAULT, VP buffers deleted, due to pool contraction
 XXXXBPuu INTEGER NOT NULL WITH DEFAULT, currently not stealable, VP buffers
 XXXXHPSz INTEGER NOT NULL WITH DEFAULT, HP size
 XXXXHPal INTEGER NOT NULL WITH DEFAULT, HP buffers allocated in active HP
 XXXXHPDE INTEGER NOT NULL WITH DEFAULT, HP buffers deleted, due to pool contraction
XXXHPBE INTEGER NOT NULL WITH DEFAULT, HP buffers backed by expanded storage
XXXVPSQ SMALLINT NOT NULL WITH DEFAULT, VPSEQ; VP sequential steal threshold
XXXHPSQ SMALLINT NOT NULL WITH DEFAULT, HP sequential steal threshold
XXXDFWR SMALLINT NOT NULL WITH DEFAULT, free buffer DWTH
XXXVDFW SMALLINT NOT NULL WITH DEFAULT, VDWT for VP
XXXIOPS SMALLINT NOT NULL WITH DEFAULT) VPPSEQ, sequential steal threshold for parallel I/O
IN XXXX05D XXXX81S ;
CREATE TABLE TØ0DSNB45XI
(XXXXOBNM CHAR(17) NOT NULL, pageset name
XXXXOBPN SMALLINT NOT NULL, pageset partition
XXXXSTME CHAR(15) NOT NULL,
XXXXOBUC SMALLINT NOT NULL, pageset use count
XXXXBPNM CHAR(4) NOT NULL, bufferpool-id
XXXXCCAC INTEGER NOT NULL WITH DEFAULT, current cached pages
XXXXMCAC INTEGER NOT NULL WITH DEFAULT, max cached pages
XXXXCHNG INTEGER NOT NULL WITH DEFAULT, current changed pages
XXXXMCHG INTEGER NOT NULL WITH DEFAULT) max changed pages
IN XXXX05D XXXX82S ;
CREATE TABLE TØ9DSNB455I
(XXXXOBNM CHAR(17) NOT NULL,
XXXXOBPN SMALLINT NOT NULL,
XXXXSTME CHAR(15) NOT NULL,
XXXXOBUC SMALLINT NOT NULL,
XXXXBPNM CHAR(4) NOT NULL,
XXXXSADL INTEGER NOT NULL WITH DEFAULT, average sync I/O delays
XXXXSMDL INTEGER NOT NULL WITH DEFAULT, max sync I/O delay
XXXXTPGS INTEGER NOT NULL WITH DEFAULT) total pages read or written
IN XXXX05D XXXX83S ;
CREATE TABLE T1ØDSNB456I
(XXXXOBNM CHAR(17) NOT NULL,
XXXXOBPN SMALLINT NOT NULL,
XXXXSTME CHAR(15) NOT NULL,
XXXXOBUC SMALLINT NOT NULL,
XXXXBPNM CHAR(4) NOT NULL,
XXXXAADL INTEGER NOT NULL WITH DEFAULT, average async I/O delay
XXXXAMDL INTEGER NOT NULL WITH DEFAULT, max async I/O delay
XXXXTPGS INTEGER NOT NULL WITH DEFAULT, total pages read or written
XXXXTIOS INTEGER NOT NULL WITH DEFAULT) total number of I/Os
IN XXXX05D XXXX83S ;

EXAMPLE LOAD STATEMENT
LOAD DATA INDDN SYSRECØØ RESUME YES INTO TABLE TØ4DSNB42XI WHEN (1:2) = 'Ø6'
( XXXXBPNM POSITION( 3 ) CHAR(4) ).
SOURCE CODE

PALØP:PROC OPTIONS(MAIN);
%INCLUDE PLINIT;
DCL IN FILE RECORD SEQUENTIAL INPUT
  ENV(TOTAL,VB,RECSIZE(137));
DCL OUT FILE RECORD SEQUENTIAL OUTPUT
  ENV(TOTAL,FB,RECSIZE(300));
DCL IN_REC           CHAR(137)       INIT('');
DCL TEMP_REC         CHAR(137)       INIT('');
DCL OUT_REC          CHAR(240)       INIT('');
DCL TIMEX            CHAR(11)        INIT('');
DCL COMMA            CHAR(1)         INIT(',');
DCL ALPHABET         CHAR(27)        INIT(',ABCDEFGHIJKLMNOPQRSTUVWXYZ');
DCL BLNKABET         CHAR(27)        INIT('                           ');
DCL DATETIME         CHAR(15)        INIT('');
DCL (DATE,TIME,ADDR,SUBSTR,INDEX,TRANSLATE)   BUILTIN;
DCL EOF              BIT(1)          INIT('0'B);
DCL (I,J)            BIN FIXED(15,Ø) INIT(Ø);  
DCL CHK845Ø          BIN FIXED(15,Ø) INIT(Ø);
DCL 1 OUT_41Ø,
  2 CUMTIME        CHAR(25);
DCL 1 OUT_42X,
  2 TYPE42X        CHAR(2),
  2 BP42X          CHAR(4),
  2 FILL42XØ       CHAR(1),
  2 RTIME42X       CHAR(15),
  2 FILL42X1       CHAR(1),
  2 TIME42X        CHAR(25),
  2 FILL42X2       CHAR(1),
  2 SYSPGUPD       CHAR(12),
  2 FILL42X3       CHAR(1),
  2 SYSPGWR        CHAR(12),
  2 FILL42X4       CHAR(1),
  2 ASYNCWIO       CHAR(12),
  2 FILL42X5       CHAR(1),
DCL 1 OUT_43Y,
  2 TYPE43Y  CHAR(2),
  2 BP43Y   CHAR(4),
  2 FILL43YØ CHAR(1),
  2 RTIME43Y CHAR(15),
  2 FILL43Y1 CHAR(1),
  2 TIME43Y CHAR(25),
  2 FILL43Y2 CHAR(1),
  2 NSYNCRD CHAR(12),
  2 FILL43Y3 CHAR(1),
  2 NSYNCWR CHAR(12),
  2 FILL43Y4 CHAR(1),
  2 NASYNCRD CHAR(12),
  2 FILL43Y5 CHAR(1),
  2 NASYNCWR CHAR(12),
  2 FILL43Y6 CHAR(1),
  2 NRDFAIL CHAR(12),
  2 FILL43Y7 CHAR(1),
  2 NWRFAIL CHAR(12),
  2 FILL43Y8 CHAR(1),
  2 UREADS CHAR(12),
  2 FILL43Y9 CHAR(1),
  2 UWRITES CHAR(12),
  2 FILL43Y1Ø CHAR(1),
  2 URFFAIL CHAR(12),
  2 FILL43Y11 CHAR(1),
  2 UWRFAIL CHAR(12),
  2 FILL43Y12 CHAR(1),
  2 IOPREQST CHAR(12),
  2 FILL43Y13 CHAR(1),
  2 IOPDEGRAD CHAR(12),
  2 FILL43Y14 CHAR(96);

DCL 1 OUT_41X,
  2 TYPE41X  CHAR(2),
  2 BP41X   CHAR(4),
  2 FILL41XØ CHAR(1),
  2 RTIME41X CHAR(15),
  2 FILL41X1 CHAR(1),
  2 TIME41X CHAR(25),
  2 FILL41X2 CHAR(1),
  2 RGETPAGE CHAR(12),
  2 FILL41X3 CHAR(1),
2 RSYNCIO       CHAR(12),
2 FILL41X4      CHAR(1),
2 SGETPAGE      CHAR(12),
2 FILL41X5      CHAR(1),
2 SSYNCIO       CHAR(12),
2 FILL41X6      CHAR(1),
2 DMTHHIT       CHAR(12),
2 FILL41X7      CHAR(1),
2 SPREQST       CHAR(12),
2 FILL41X8      CHAR(1),
2 SPIOs         CHAR(12),
2 FILL41X9      CHAR(1),
2 SPPGREAD      CHAR(12),
2 FILL41X10     CHAR(1),
2 LPREQST       CHAR(12),
2 FILL41X11     CHAR(1),
2 LPIOs         CHAR(12),
2 FILL41X12     CHAR(1),
2 LPPGREAD      CHAR(12),
2 FILL41X13     CHAR(1),
2 DPREQST       CHAR(12),
2 FILL41X14     CHAR(1),
2 DPIOS         CHAR(12),
2 FILL41X15     CHAR(1),
2 DPPGREAD      CHAR(12),
2 FILL41X16     CHAR(1),
2 PFNOBUF       CHAR(12),
2 FILL41X17     CHAR(1),
2 PFNORDE       CHAR(12),
2 FILL41X18     CHAR(44);

DCL 1 OUT_40X,
2 TYPE40X       CHAR(2),
2 BPNAME        CHAR(4),
2 FILL40X0      CHAR(1),
2 RTIME40X      CHAR(15),
2 FILL40X1      CHAR(1),
2 BPUCNT        CHAR(12),
2 FILL40X2      CHAR(1),
2 BPSIZE        CHAR(12),
2 FILL40X3      CHAR(1),
2 BPALLOC       CHAR(12),
2 FILL40X4      CHAR(1),
2 BPDELETE      CHAR(12),
2 FILL40X5      CHAR(1),
2 BPUSEUPD      CHAR(12),
2 FILL40X6      CHAR(1),
2 HPSIZE        CHAR(12),
2 FILL40X7      CHAR(1),
2 HPALLOC       CHAR(12),
2 FILL40X8      CHAR(1),
2 FILL40X9      CHAR(1),
2 FILL40X10     CHAR(1),
2 FILL40X11     CHAR(1),
2 FILL40X12     CHAR(1),
2 FILL40X13     CHAR(1),
2 FILL40X14     CHAR(1),
2 FILL40X15     CHAR(1),
2 FILL40X16     CHAR(1),
2 FILL40X17     CHAR(1),
2 FILL40X18     CHAR(44);
2 HPDELETE       CHAR(12),
2 FILL4ØX9       CHAR(1),
2 HPCOUNT       CHAR(12),
2 FILL4ØX1Ø      CHAR(1),
2 VPSEQN         CHAR(3),
2 FILL4ØX11      CHAR(1),
2 HPSEQN        CHAR(3),
2 FILL4ØX12      CHAR(1),
2 DEFRDWR        CHAR(3),
2 FILL4ØX13      CHAR(1),
2 VDEFRDWR       CHAR(3),
2 FILL4ØX14      CHAR(1),
2 IOPSEQN       CHAR(3),
2 FILL4ØX15      CHAR(141);

DCL 1 OUT_45X,
  2 NAME45X        CHAR(17),
  2 FILL45X_1      CHAR(1),
  2 USECOUNT       CHAR(5),
  2 FILL45X_2      CHAR(1),
  2 PARTNUM        CHAR(2);

DCL 1 OUT_453,
  2 TYPE453        CHAR(2),
  2 BP453         CHAR(4),
  2 FILL453_Ø      CHAR(1),
  2 RTIME453      CHAR(15),
  2 FILL453_1     CHAR(1),
  2 OUT_45X_3     CHAR(26),
  2 FILL453_2     CHAR(1),
  2 CCACHED       CHAR(12),
  2 FILL453_3     CHAR(1),
  2 MCACHED       CHAR(12),
  2 FILL453_4     CHAR(1),
  2 CHANGED       CHAR(12),
  2 MCHANGED      CHAR(12),
  2 FILL453_6     CHAR(199);

DCL 1 OUT_45N,
  2 TYPE45N        CHAR(2),
  2 BP45N         CHAR(4),
  2 FILL45N_Ø      CHAR(1),
  2 RTIME45N      CHAR(15),
  2 FILL45N_1     CHAR(1),
  2 OUT_45X_N     CHAR(26),
  2 FILL45N_2     CHAR(1),
  2 AVGDELAY      CHAR(12),
  2 FILL45N_3     CHAR(1),
  2 MAXDELAY      CHAR(12),
  2 FILL45N_4     CHAR(1),
  2 TOTPGS        CHAR(12),
  2 FILL45N_5     CHAR(1),
2 TOTIOS         CHAR(12),
2 FILL45N_6      CHAR(199):
ON ENDFILE(IN) EOF='1'B;
OPEN FILE(IN);
OPEN FILE(OUT);
READ FILE(IN) INTO(IN_REC);
DATETIME=DATE()||TIME();
DO WHILE(¬EOF):
  I=INDEX(IN_REC,'DSNB4Ø1I');
  IF I>Ø THEN DO:
    OUT_4ØX='';
    RTIME4ØX=DATETIME;
    J=INDEX(IN_REC,'BUFFERPOOL NAME');
    BPNAME=SUBSTR(IN_REC,J+16,4);
    J=INDEX(BPNAME,'',');
    SUBSTR(BPNAME,J,1)=' ';
    TYPE4ØX='Ø1';
    J=INDEX(IN_REC,'USE COUNT');
    BPUCNT=SUBSTR(IN_REC,J+10,5);
    BPUCNT=ZEROIT(BPUCNT,5);
    IN_REC='';
    READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB4Ø2I MESSAGE */
    J=INDEX(IN_REC,'=');
    BPSIZE=SUBSTR(IN_REC,J+1,12);
    BPSIZE=TRANSLATE(BPSIZE,BLNKABET,ALPHABET);
    BPSIZE=ZEROIT(BPSIZE,12);
    IN_REC='';
    READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB4Ø2I MESSAGE */
    J=INDEX(IN_REC,'=');
    BPALLOC=SUBSTR(IN_REC,J+1,12);
    BPALLOC=ZEROIT(BPALLOC,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+137-J);
    J=INDEX(TEMP_REC,'=');
    BPDELETE=SUBSTR(TEMP_REC,J+1,12);
    BPDELETE=ZEROIT(BPDELETE,12);
    IN_REC='';
    READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB4Ø2I MESSAGE */
    J=INDEX(IN_REC,'=');
    BPUSEUPD=SUBSTR(IN_REC,J+1,12);
    BPUSEUPD=ZEROIT(BPUSEUPD,12);
    IN_REC='';
    READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB4Ø3I MESSAGE */
    J=INDEX(IN_REC,'=');
    HPSIZE=SUBSTR(IN_REC,J+1,12);
    HPSIZE=TRANSLATE(HPSIZE,BLNKABET,ALPHABET);
    HPSIZE=ZEROIT(HPSIZE,12);
    IN_REC='';
    READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB4Ø3I MESSAGE */
    J=INDEX(IN_REC,'=');
HPALLOC=SUBSTR(IN_REC,J+1,12);
HPALLOC=ZEROIT(HPALLOC,12);
TEMP_REC='';
TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
J=INDEX(TEMP_REC,'=');
HPDELETE=SUBSTR(TEMP_REC,J+1,12);
HPDELETE=ZEROIT(HPDELETE,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB403I MESSAGE */
J=INDEX(IN_REC,'=');
HPBACKES=SUBSTR(IN_REC,J+1,12);
HPBACKES=ZEROIT(HPBACKES,12);
READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB404I MESSAGE */
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB404I MESSAGE */
J=INDEX(IN_REC,'=');
VPSEQN=SUBSTR(IN_REC,J+1,3);
VPSEQN=ZEROIT(VPSEQN,3);
TEMP_REC='';
TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
J=INDEX(TEMP_REC,'=');
HPSEQN=SUBSTR(TEMP_REC,J+1,3);
HPSEQN=ZEROIT(HPSEQN,3);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB404I MESSAGE */
J=INDEX(IN_REC,'=');
DEFRDWR=SUBSTR(IN_REC,J+1,3);
DEFRDWR=ZEROIT(DEFRDWR,3);
TEMP_REC='';
TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
J=INDEX(TEMP_REC,'=');
VDEFRDWR=SUBSTR(TEMP_REC,J+1,3);
VDEFRDWR=ZEROIT(VDEFRDWR,3);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB404I MESSAGE */
J=INDEX(IN_REC,'=');
IOPSEQN=SUBSTR(IN_REC,J+1,3);
IOPSEQN=ZEROIT(IOPSEQN,3);
OUT_REC='';
WRITE FILE(OUT) FROM(OUT_40X);
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB450I')+INDEX(IN_REC,'DSNB451I');
IF I>Ø THEN DO;
OUT_45X='';
J=INDEX(IN_REC,'=');
NAME45X=SUBSTR(IN_REC,J+2,17);
I=INDEX(NAME45X,'.');
IF I>Ø THEN NAME45X=SUBSTR(NAME45X,1,I-1);
TEMP_REC='';
TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
J=INDEX(TEMP_REC,'=');
USECOUNT=SUBSTR(TEMP_REC,J+1,5);
USECOUNT=ZEROIT(USECOUNT,5);
IN_REC='';
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB452I');
IF I>Ø THEN DO:
   J=INDEX(IN_REC,'DATASET');
   PARTNUM=SUBSTR(IN_REC,J+8,2);
   PARTNUM=ZEROIT(PARTNUM,2);
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB453I');
IF I>Ø THEN DO:
   OUT_453='';
   TYPE453='Ø2';
   BP453=BPNAME;
   OUT_45X_3=NAME45X||FILL45X_1||USECOUNT||FILL45X_2||PARTNUM;
   RTIME453=DATETIME;
   IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB453 MESSAGE */
   J=INDEX(IN_REC,'=');
   CCACHED=SUBSTR(IN_REC,J+1,12);
   CCACHED=TRANSLATE(CCACHED,BLNKABET,ALPHABET);
   CCACHED=ZEROIT(CCACHED,12);
   TEMP_REC='';
   TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
   J=INDEX(TEMP_REC,'=');
   MCACHED=SUBSTR(TMP_REC,J+1,12);
   MCACHED=TRANSLATE(MCACHED,BLNKABET,ALPHABET);
   MCACHED=ZEROIT(MCACHED,12);
   IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB453 MESSAGE */

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

M K Mohan
DB2 Specialist (UK) © Xephon 1998
Legato Systems has announced NetWorker BusinessSuite Module for DB2, providing on-line back-up for DB2 Universal Database running on AIX, and providing the ability to manage and back up multiple DB2 servers centrally. This follows joint development work with IBM.

The new module automates the back-up process, increases data availability, and provides disaster recovery support. Other features include the ability to perform a recovery down to the tablespace level, coupled with lights-out operation via built-in scheduling and tape library support.

Database administrators can back up all DB2 files while creating a secondary copy for off-site storage, and integrate application back-ups with filesystem back-ups.

For further information contact:
Legato Systems, 3210 Porter Drive, Palo Alto, CA 94304, USA.
Tel: (650) 812 6200.

IBM has announced further details of The DB2 Universal Database Server for OS/390 Version 6 (DB2 Update, July 1998).

The universal database allows users to store and query not only alphanumeric data but also text documents, images, audio, video, and other complex objects. With Version 6, it is possible to take advantage of the UDB object/relational capabilities across IBM and non-IBM operating systems.

Enhancements include performance improvements for utilities, faster restart and recovery, better query performance, greater data capacity, and more built-in functions.

Support for complex data types and LOBs (large objects) comes via DB2 Extenders. A LOB column can be up to 2GB and a collection of all LOB values can be up to 4,000TB. Each extender is a package of predefined UDTs, UDFs, triggers, constraints, and stored procedures.

New tools in Version 6 include QMF and QMF for Windows, DB2 DataPropagator, DB2 Administration Tool, and DB2 Buffer Pool Tool.

For further information contact your local IBM representative.

DB2 security data can now be administered in the same way as RACF data using Version 3.1.2 of BETA 88, BETA Systems’ host-based Systems’ Enterprise Security Manager for RACF administration. BETA 88 can be used to view, modify, query, and generate reports from any security-related DB2 data covered by RACF.

For further information contact:
BETA Systems Software, One Securities Center, 3490 Piedmont Road, Suite 1100, Atlanta, GA 30305, USA.
Tel: (404) 812 1556.
BETA Systems Software, Highlands House, Basingstoke Road, Spencers Wood, Reading, RG7 1NT, UK.
Tel: (01734) 885175.