June 2000

In this issue

3 Capturing accounting information
15 DB2 Version 5 catalog statistics
17 Using LSTCAT output to generate ALTER SQL statements
23 DB2 PLAN_TABLE – access and maintenance
32 Image copy, DSNTIAUL copy, and disaster recovery of DB2 objects – part 2
48 DB2 news

© Xephon plc 2000
 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 userid shown on your address label.

 © Xephon plc 2000. 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.
Capturing accounting information

In order to tune application programs, many DBAs are using a third-party monitoring program. If there is no monitoring tool, SMF data should be used. The process of handling SMF data is complicated.

This ALC program gives accounting information in a real-time environment – like a monitoring tool.

The program starts a trace with an Online Performance (OP) destination, and captures data from OP buffers, then formats the report.

Note:

- The user must have DB2 TRACE authority.
- Because the report is printed in a JES spool, you must be careful if you have a heavily-used DB2 system.
- If other programs are using the OP buffer, you need to control the OP buffers.
- To stop the batch job, you must issue an MVS cancel command.
- To stop the previously invoked trace, you must issue the ‘-sto trace(a) tno(#)’ DB2 command.

Example output is shown in Figure 1.

ASEMBLE AND EXECUTING JCL

```
//JOBLIB DD DSN=DB2510.SDSNLOAD,DISP=SHR
//*
//PREPUNL EXEC DSNHASM, MEM=DB2CPACT,
//    PARM.PC='HOST(ASM),STDSQL(NO)',
//    PARM.ASM='OBJECT,NODECK',
//    PARM.LKED=(MAP,LET,LIST)
//PC.DBRMLIB DD DSN=DB2T.DBRMLIB(DB2CPACT),
//    DISP=SHR
//PC.SYSLIB DD DSN=DB2510.SDSNSAMP,
//    DISP=SHR
//PC.SYSIN DD DSN=DB2T.ASMLIB(DB2CPACT),
//    DISP=SHR
//ASM.SYSLIB DD
//    DD DSN=DB2510.SDSNMACS,
```
Figure 1: Example output

//
// DISP=SHR
//LKED.SYSLMOD DD DSN=DB2T.LOADLIB(DB2CPACT),
// DISP=SHR
//LKED.SYSIN DD *
// INCLUDE SYSLIB(DSNELI)
NAME DB2CPACT(R)
/**
DB2CPACT

- PROGRAM SOURCE CODE

TITLE 'DB2 ACCOUNTING CAPTURE PROGRAM'

* DB2CPACT : REAL-TIME ACCOUNTING INFORMATION CAPTURE *
* FUNCTION : *
* - START ACCOUNTING TRACE IN PROGRAM *
* - CAPTURE ACCOUNTING INFORMATION FROM 'ONLINE PERFORMANCE BUFFER' *
* - FORMATTING REPORT *
* ** NOTE) *
* - YOU MUST CHANGE 'OWNR' FIELD WITH YOUR LOGON ID *
* AND HAVE '-STA TRACE' DB2 AUTHORITY. *
* - ONLINE PERFORMANCE BUFFER (OP1) IS USED BY THIS PROGRAM. *
* - IF YOU WILL USE A SPECIFIC OP#, CHANGE IT. *
* - THIS IS A LOOPING PROGRAM. TO STOP, YOU MUST CANCEL THIS JOB. *
* AND YOU MUST ISSUE THE '-STO TRACE' COMMAND TO STOP THE TRACE. *
* - SOME OUTPUT FIELDS ARE IN HEX FORMAT.
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
* PSEUDOCODE
* OPEN FILES
* WRITE REPORT HEADER OUTPUT
* START TRACE
* . USE GETMAIN TO OBTAIN A STORAGE SAME WITH BUFSIZE
* . START TRACE WITH DEST=OPX
* . INDICATE TO WAKE UP THIS ROUTINE BY A POST
* . WHENEVER THE BUFFER IS 20% FULL
* . WAIT FOR THE BUFFER TO BE POSTED
* READ TRACED DATA FROM BUFFER
* . CALL IFI TO OBTAIN THE BUFFER DATA VIA A READA REQUEST
* . CHECK THE STATUS IF THE READA WAS SUCCESSFUL
* FORMAT THE OUTPUT DATA
* LOOP BACK TO THE WAIT
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
* REGISTERS
* R2          IFI RETURN AREA
* R3   IFCA   INSTRUMENTATION FACILITY COMMUNICATION AREA
* R4   QWAØ   SELF DEFINE SECTION
* R5   QWHS   PRODUCTION SECTION - STANDARD HEADER
* QWHC   PRODUCTION SECTION - CORRELATION HEADER
* QWAC   DATA SECTION
* QXST   SQL SECTION
* QBAC   BUFFER MANAGER SECTION
* QPAC   BUFFER MANAGER SECTION
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
* DD CARDS     :
* SYSPRINT - MESSAGE DD
* REPORT - OUTPUT DD
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
* MACROS       :
* DSNDIFCA      IFCA MAPPING MACRO
* DSNDWBUF      IFC BUFFER INFORMATION BLOCK
* DSNDWQAL      IFC QUALIFICATION BLOCK
* DSNDQWAS      ACCOUNTING MAPPING MACRO
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
* OUTPUT FIELD :
* 1) PLAN INFORMATION :
* QWHCAID - AUTHORIZATION ID
* QWHCCV - CORRELATION ID
* QWHCCN - CONNECTION NAME
* QWHCPLAN - PLAN NAME
* QWACASC - ELAPSED TIME IN DB2
* QWACAJST - TCB TIME IN DB2
* QWACAWTI - I/O WAIT TIME
* QWACAWTR - ASYNCH READ WAIT TIME
* QWHSCLUCC - COMMIT COUNT (HEX OUTPUT)
* QXSELECT - # OF SELECT SQL
* QXINSRT - # OF INSERT SQL
* QXUPDTE - # OF UPDATE SQL
* QXDELET - # OF DELETE SQL
* QXFETCH - # OF FETCH SQL
* TOTGET - TOTAL GET PAGE COUNT

2) BUFFER INFORMATION

* QBACPID - BUFFER POOL ID
* QBACGET - # OF GET PAGE (HEX OUTPUT)
* QBACRIO - # OF SYNCRONOUS I/O (HEX OUTPUT)
* QBACSEQ - # OF SEQUENTIAL PREFETCH I/O (HEX OUTPUT)
* QBACLPF - # OF LIST PREFETCH I/O (HEX OUTPUT)
* QBACDPF - # OF DYNAMIC PREFETCH I/O (HEX OUTPUT)

3) PACKAGE INFORMATION

* QPACRECN - # OF PACKAGE OR DBRM
* QPACLOCN - LOCATION NAME
* QPACCOLN - COLLECTION ID
* QPACPKID - PACKAGE NAME
* QPACCONT - CONSISTENCY TOCKEN
* QPACSCST - PACKAGE ELAPSE TIME
* QPACAWTI - PACKAGE WAIT TIME

REFERENCE:

'ADMINISTRATION GUIDE' APPENDIX E. PROGRAMMING FOR THE IFI

space
DB2CPACT CSECT
SAVE (14,12)
LR R12,R15
USING DB2CPACT,R12
ST R13,SAVE+4
LA R8,SAVE
ST R8,8(R13)
LR R13,R8

*--- OPEN DD ---*
OPEN (OUTPUT1,(OUTPUT),REPORT,(OUTPUT))

*--- WRITE REPORT HEADER ---*
BAL R14,HEADRTN

*--- START TRACE ---*
BAL R14,STATRAC

*--- READ TRACE AND FORMAT REPORT ---*
BAL R14,READRTN

*--- IF YOU DON'T WANT TO LOOP PGM, THE FOLLOWINGS ARE USED. ---*
CLOSRTN DS ØH
CLOSE (OUTPUT1,REPORT)
L R13,SAVE+4
L R15.=F'8'
RETURN (14,12),RC=(15)
*--- ISSUE TRACE COMMAND ---*

STATRAC DS $0H
ST R14,STATSAVE

*--- IFCA AREA INITIALIZATION ---*

LA R2,IFCAAREA
USING IFCA,R2
MVC IFCALEN(2),LENIFCA          MOVE IFCA LENTH
MVC IFCAID(4),IFCAEYE           MOVE CHARACTER 'IFCA'
MVC IFCAOWNR(4),OWNR            MOVE 'OWNR'
LA R2,BUFAREA
USING WBUF,R2
MVC WBUFLEN(2),LENWBUF          MOVE WBUF LENTH
MVC WBUFENEYE(4),WBUFENEYE1     MOVE CHARACTER 'WBUF'
MVC WBUFECB(4),ECB1ADDR         EDB ADDRESS
MVC WBUFBC(4),BUFCT
DROP R2
GETMAIN EC,LV=8192,A=STOADDR,LOC=BELOW
L R2,STOADDR
A R2,=F'8'
ST R2,RETADDR
MVC $0(4,R2),=F'4Ø88'

*--- ISSUE START TRACE ---*
CALL DSNWLI,(COMMAND,IFCAAREA,(R2),OUTAREA,BUFAREA),VL

*--- CHECK RETURN AND REASON CODE ---*

LA R3,IFCAAREA
USING IFCA,R3
CLC IFCARC1,=F'0'           CHECK RETURN CODE
BE PONGØ
MVC OUTDATA(4),IFCARC1

PONGØ CLC IFCARC2,=F'0'
BE STATRXT
MVC OUTDATA+4(4),IFCARC2
MVC OUTDATA+10(24),=C': CHECK LEFT REASON CODE'
PUT OUTUT1,OUTDATA
B CLOSRTN

STATRXT DS $0H
DROP R3
MVC OUTDATA(255),RETADDR
PUT OUTUT1,OUTDATA
L R14,STATSAVE
BR R14
SPACE

*--- ---*

READRTN DS $0H
WAIT ECB=ECB1
MVC ECB1(4),=F'0'          CLEAR POST FLAG

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 410 0290.
L R2,RETADDR
MVC Ø(4,R2),=F'8184' BUFFER SIZE
*---
CALL DSNWLI,(READA,IFCAAREA,(R2)),VL
*---
CHECK RETURN AND RESON CODE
LA R3,IFCAAREA
USING IFCA,R3
CLC IFCARC1,=F'Ø'
BE PONG1
MVC OUTDATA(4),IFCARC1
PONG1 CLC IFCARC2,=F'Ø'
BE READRXT
MVC OUTDATA+4(4),IFCARC2
MVC OUTDATA+8(ØØ),IFCABM
PUT OUTUT1,OUTDATA
B CLOSRTN
READRXT DS ØH
MVC Ø(2,R2),IFCABM+2
DROP R3
MVC 2(2,R2),=H'Ø'
BAL R14,REPTRTN
B READRTN

*--- FORMATTING REPORT
REPTRTN DS ØH
ST R14,REPTSAVE
LR R4,R2
LA R4,8(R4) SKIP LENGTH BYTE 4
USING QWAØ,R4
LR R5,R4
ST R5,OFFSET SAVE OFFSET POINT
A R5,QWAØ1PSO STANDARD HEADER OFFSET
S R5,=F'4'
USING QWHS,R5
CLI QWHSTYP,QWHSØ1 IS TYPE STANDARD ?
BNE REPTRXT
*
LH R11,QWHSLUCC COMMIT COUNT
BAL R14,CVDRTN CONVERSION TO ZONE DECIMAL
MVC QWHSLUCC,PACK1+12
*
AH R5,QWHSLEN
USING QWHC,R5
CLI QWHCTYP,X'2' IS IT CORRELATION SECTION?
BNE REPTRXT
CLC QWHCAID,=C'SYSPR ' IF SYSTEM PLAN?
BE REPTRXT SKIP.
MVC OWHCAID,OWHCAID AUTH ID
MVC OWHCCV,OWHCCV CORRELATION ID
MVC OWHCCN,OWHCCN CONNECTION NAME
MVC OWHCPLAN,QWHCPLAN PLAN NAME
DROP R5
CLC OWHCPLAN,=C'DB2CPACT' IF MY PROGRAM?
BE REPTRXT SKIP.
CLC OWHCPLAN,=C' IF PLANNAME US NULL?
BNE PING NO, GO AHEAD
CLC QWA01R5O,=X'00000000' IS IT DDF PROGRAM?
BE REPTRXT IF DDF SECTION IS NULL
* GO END

PING L R5,OFFSET
A R5,QWA01R1O CORRELATION OFFSET
S R5,=F'4'
USING QWAC,R5 CORRELATION SECTION
*
MVC TODFROM,QWACASC MOVE DB2 E-TIME
BAL R14,CONVRTN
MVC QWACASC,MILITIME BINARY DB2 ELAPSED TIME SSS.SS
*
MVC TODFROM,QWACAJST MOVE TCB TIME
BAL R14,CONVRTN
MVC QWACAJST,MILITIME BINARY DB2 CPU TIME SSS.SS
*
MVC TODFROM,QWACAWTI MOVE WAIT TIME
BAL R14,CONVRTN
MVC QWACAWTI,MILITIME BINARY DB2 I/O WAIT TIME
*
MVC TODFROM,QWACAWTR MOVE ASYNCH READ WAIT TIME
BAL R14,CONVRTN
MVC QWACAWTR,MILITIME ASYNCH READ I/O WAIT TIME
DROP R5
CLC QWA01R2O,=X'00000000' IS IT NULL FUNCTION?
BNE NEXTØ IF YES, SKIP
MVC OXSELECT,=C'0' INITIALIZE
MVC OXINSRT,=C'0'
MVC OXUPDTE,=C'0'
MVC OXDELETE,=C'0'
MVC OXFETCH,=C'0'
MVC OXGETPG,=C'0'
B NEXT1 IF YES, SKIP

NEXTØ
DS 0H
L R5,OFFSET SET INITIAL POINT
A R5,QWA01R2O OFFSET OF SQL SECTION
S R5,=F'4'
USING QXST,R5
*
L R11,QXSELECT # OF SELECT STATEMENT
BAL R14,CVDRTN
MVC OXSELECT,PACK1+8
*
L R11,QXINSRT # OF INSERT STATEMENT
BAL R14,CVDRTN

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 410 0290.
MVC OXINSRT,PACK1+8
* # OF UPDATE STATEMENT
L R11,OXUPDTE
BAL R14,CVDRTN
MVC OXUPDTE,PACK1+8
*
L R11,OXDELET
BAL R14,CVDRTN
MVC OXDELET,PACK1+8
*
L R11,OXFETCH
BAL R14,CVDRTN
MVC OXFETCH,PACK1+8
DROP R5
*
* BUFFER MANAGER (DSNDQBAC) SECTION
NEXT1 DS ØH
CLC QWAØ1R3O,=X'00000000' IS IT NULL FUNCTION?
BE NEXT2 IF YES, SKIP
L R5,OFFSET SET INITIAL POINT
A R5,QWAØ1R3O OFFSET OF BUFFER MANAGER SECTION
S R5,=F'4'
LH R6,QWAØ1R3N # OF BUFFER MANAGER DATA SECTION
LA R8,REPORTO+137 POINT OF OUTAREA
SR R1Ø,R1Ø
LOOPBUFF DS ØH
USING QBAC,R5 BUFFER MANAGER DSECT
L R7,OBACPID BUFFER POOL ID
CVD R7,PACKWORK
UNPK BPNUM+2(2),PACKWORK+6(2)
OI BPNUM+3,X'FØ' CONVERT TO ZONE DECIMAL
MVC Ø(4,R8),BPNUM
*
A R1Ø,QBACGET ADD TO TOTAL GET PAGE
MVC 5(4,R8),QBACGET # OF GET PAGE
MVC 10(4,R8),QBACRIO # OF SYNC READ I/O
MVC 15(4,R8),QBACSEQ # OF SEQ PREFETCH I/O
MVC 20(4,R8),QBACLPF # OF LIST PREFETCH I/O
MVC 25(4,R8),QBACDPF # OF DYNAMIC PREFETCH I/O
*
LA R8,3Ø(R8) POINTER OF NEXT PRINT
LH R9,QWAØ1R3L LOAD LENGTH OF BUFFER SECTION
AR R5,R9 POINTER OF NEXT BUFFER SECTION
BCT R6,LOOPBUFF
NEXT2 DS ØH
ST R1Ø,TOTGET SAVE TOTAL GET PAGE
*
L R11,TOTGET
BAL R14,CVDRTN CONVERSION TO ZONE DECIMAL
MVC OXGETPG,PACK1+8
*
DROP R5
PUT REPORT,REPORTO WRITE OUTPUT
XC REPORTO,REPORTO CLEARE

*CLC QWA01R80,=X'00000000' IS IT NULL FUNCTION?
BE REPTRXT
L R5,OFFSET SET INITIAL POINT
A R5,QWA01R80 OFFSET OF BUFFER MANAGER SECTION
S R5,=F'4'
LH R6,QWA01R8N # OF BUFFER MANAGER DATA SECTION
LA R8,REPORTO+40 POINT OF OUTAREA
SR R10,R10 CLEAR R5

LOOPPACK DS ØH
USING QPAC,R5
MVC REPORTO+1(35),=C'** PACKAGE OR DBRM OF ABOVE PLAN **'
MVC Ø(2,R8),QPACRECN # OF PACKAGE
MVC 2(16,R8),QPACLOCN LOCATION ID
MVC 18(18,R8),QPACCOLN COLLECTION ID
MVC 36(18,R8),QPACPKID PACKAGE NAME
MVC 54(11,R8),=C'CON-TOCKEN:'
MVC 65(8,R8),QPACCONT CONSISTENCY TOKEN

*MVC 73(10,R8),=C'DB2-ETIME:'
MVC TODFROM,QPACSCT MOVE PACKAGE E-TIME
BAL R14,CONVRTN
MVC 83(11,R8),MILITIME BINARY DB2 ELAPSED TIME SSS.SS

*MVC 95(10,R8),=C'DB2-WTIME:'
MVC TODFROM,QPACWINTI MOVE PACKAGE W-TIME
BAL R14,CONVRTN
MVC 105(11,R8),MILITIME BINARY DB2 ELAPSED TIME SSS.SS

*LH R9,QWA01R8L LOAD LENGTH OF BUFFER SECTION
AR R5,R9 POINTER OF NEXT BUFFER SECTION
PUT REPORT,REPORTO WRITE OUTPUT
XC REPORTO,REPORTO CLEAR
BCT R6,LOOPPACK LOOPING

REPTRXT DS ØH
DROP R4
L R14,REPTSAVE
BR R14

.NET CONVERT TIME FORMAT TO DISPLAY ---*
CONVRTN DS ØH
ST R14,CONVSAVE
STCKCONV STCKVAL=TODFROM,CONVVAL=TODTO,
X TIMETYPE=BIN,DATETYPE=YYYYDDD
UNPK TOTIME(9),TODTO(5)
MVC MILITIME(2),TODTIME
MVC MILITIME+3(2),TODTIME+2
MVC MILITIME+6(2),TODTIME+4
MVC MILITIME+9(2),TODTIME+6

CONVRXT DS ØH
L R14,CONVSAVE
BR R14
SPACE

*--- CONVERT HEX TO ZONE DECIMAL ----*

CVDRTN DS ØH
ST R14,CVDSAVE
CVD R11,PACKWORK
MVC PACK1,EDIT2
ED PACK1,PACKWORK

CVDRXT DS ØH
L R14,CVDSAVE
BR R14
SPACE

*--- HEADER ROUTINE ----*

HEADRTN DS ØH
ST R14,HEADSAVE
MVC REPORTO+Ø(8),=C'*AUTHID/' AUTHORIZATION ID
MVC REPORTO+8(12),=C'* CORR-ID /' CORRELATION ID
MVC REPORTO+20(8),=C'*CONNID/' CONNECTION ID
MVC REPORTO+28(8),=C'*PLAN*/' PLAN NAME
MVC REPORTO+36(11),=C'*E-TIME**/' ELAPSE TIME
MVC REPORTO+47(11),=C'*C-TIME**/' CPU TIME
MVC REPORTO+58(11),=C'*WAIT I/O*/' WAIT I/O TIME
MVC REPORTO+69(11),=C'*WAIT I/O2/' WAIT WRITE TIME
MVC REPORTO+80(9),=C'*COMMIT*/' COMMIT COUNT
MVC REPORTO+89(8),=C'*SELECT/' SELECT COUNT
MVC REPORTO+97(8),=C'*INSERT/' INSERT COUNT
MVC REPORTO+105(8),=C'*UPDATE/' UPDATE COUNT
MVC REPORTO+113(8),=C'*DELETE/' DELETE COUNT
MVC REPORTO+121(8),=C'*FETCH*/' FETCH COUNT
MVC REPORTO+129(8),=C'*GETPG*/' TOTAL GETPAGE COUNT
MVC REPORTO+137(4),=C'*BP/' BUFFER POOL NAME
MVC REPORTO+142(4),=C'*GP/' GETPAGE / BP
MVC REPORTO+147(4),=C'*SR/' SYNC READ I/O COUNT
MVC REPORTO+152(4),=C'*SP/' SEQUENTIAL PREFETCH
MVC REPORTO+157(4),=C'*LP/' LIST PREFETCH
MVC REPORTO+162(4),=C'*DP/' DYNAMIC PREFETCH
MVC REPORTO+168(27),=C'(*: START AND /: END POINT)'
PUT REPORT,REPORTO
XC REPORTO,REPORTO

HEADRXT DS ØH
L R14,HEADSAVE
BR R14
SPACE

*--- COMMAND ----*

COMMAND DC CLB'COMMAND '
READA DC CLB'READA '

*--- STORAGE OF LENGTH(IFCA) AND PROPERLY INITIALIZED
LENIFCA DC AL2(AFTIFCA-IFCA)
IFCAEYE DC 'IFCA'
OWNR DC 'XDBJ'
IFCAAREA DC XL(AFTIFCA-IFCA)'Ø'
* --- STORAGE OF LENGTH(WBUF) AND PROPERLY INITIALIZED
LENWBUF DC AL2(AFTWBUF-WBUF)
WBUFYEYE DC 'WBUF'
BUFAREA DC XL(AFTWBUF-WBUF)'Ø'
* --- STORAGE FOR LENGTH AND RETURNED INFO
RETTADDR DS A
STOADDR DS A
* --- STORAGE FOR LENGTH AND DB2 COMMAND
OUTAREA DS ØCL44
OUTLEN DC X'002C0000' MUST CHANGE WHEN COMMAND IS CHANGED
OUTCMD DC CL40'-STA TRACE(A) DEST(OPX) CLASS(1,2,3,7,8)'
SAVE DS 18F
STATSAVE DC F'Ø'
REPTSAVE DC F'Ø'
CONVSAVE DC F'Ø'
CVDSAVE DC F'Ø'
HEADSAVE DC F'Ø'
OFFSET DC F'Ø'
BUFCT DC F'20'
ECBI DC F'Ø'
ECBIAADDR DC A(ECBI)
TODFROM DS CL8 WORK AREA FOR TIME CONVERT
TODTO DS CL16
TODTIME DS CL9
TOTGET DC F'Ø' # OF TOTAL GET PAGE
PACKWORK DS PL8 TIME PACK DECIMAL
PACK1 DS XL16 BUFFER PACK DECIMAL
MILITIME DC CL11'HH:MM:SS.XX' TCB & ELAPSE TIME DISPLAY
BPNUM DC CL4'BP##' BUFFER POOL NAME
EDIT2 DC X'4020202020202020202020202120' OUTDATA DS CL256
*
REPORTO DS ØCL250
OWHCAID DS CL8
OWHCCV DS CL12
OWHCCN DS CL8
OWHCPLAN DS CL8
OWACASC DS CL11
OWACAJST DS CL11
OWACAWTI DS CL11
OWACANTR DS CL11
OWHSLUCC DS F # COMMIT POINT
OXSELECT DS D
OXINSR T DS D
OXUPDTE DS D
OXDELETE DS D
OXFETCH DS D
OXGETPG DS D
**DB2 Version 5 catalog statistics**


In Version 5, floating decimal fields were added to cater for large tables.

RUNSTATS will still update the original integer fields and the new floating point fields for non-large tables, and will update the floating decimal fields but not the corresponding integer fields for large tables. The DB2 optimizer now uses the floating point fields.

The changes here do not cater for large tables but are only intended to ensure that statistics entered are copied to the fields used by the optimizer.

The changes are straightforward and are marked in italics.
Alter comments:

/* REXX */
/*
/* This EXEC will retrieve and update catalog statistics */
/* for a given DB2 table. */
/*
/* The EXERRC has been altered for Version 5 of DB2 to update */
/* the '%CARDF' catalog statistics in line with the '%CARD' */
/* statistics. CARDF, therefore, can never be greater than */
/* CARD. */
/*

Find routine:

DB_UPDATE:
signal on failure

Alter update statement:

UPDT = "UPDATE SYSIBM.SYSTABLES",
"SET CARD=\"ROWS\",NPAGES=\"NPAGES\",",
"PCTPAGES=\"PCTPAGES\",CARDF=\"ROWS",
WHRCLS = "WHERE CREATOR='\"CRTR\'' AND NAME='\"TBNAM\'''

Alter update statement:

UPDT = "UPDATE SYSIBM.SYSINDEXES ",
"SET FIRSTKEYCARD=\"NFSKCRD\",FULLKEYCARD=\"FLKCRD\",",
"FIRSTKEYCARD=\"NFSKCRD\",FULLKEYCARD=\"FLKCRD\",",
"NLEAF=\"NLEAF\",NLEVELS=\"NLVLS\",CLUSTERRATIO=\"NCRIO",
WHRCLS = "WHERE CREATOR='\"ICRTR\'' AND NAME='\"INAME\'''

Find routine:

EA_UPDATE:

Alter update statement:

UPDT = "UPDATE SYSIBM.SYSCOLUMNS ",
"SET COLCARD=\"NCCARD\",LOW2KEY=\"NL2KEY\'',",
"HIGH2KEY=\"NH2KEY\'',COLCARD=\"NCCARD",
WHRCLS = "WHERE TBCREATOR='\"CRTR\'' AND TBNAME='\"TBNAM\'',
"AND NAME='\"UCNAME\''

Liz Page
Independent Consultant (UK) © Xephon 2000
Using LSTCAT output to generate ALTER SQL statements

The GENALTR utility works on the output of the LSTCAT utility. It checks the old PRIQTY and the new PRIQTY and, if they are different, it will generate ALTER SQL statements for that particular object, which could be an indexspace or a tablespace.

Apart from the input dataset, other input required includes:

- A creator name for the index.
- A default percentage for the secondary quantity. (This specifies the percentage of the new primary quantity to be used as the new secondary quantity.)

The input file to the GENALTR utility is coded as an argument and hence it can be conveniently executed from an ISPF 3.4 list panel containing the output of the LSTCAT execution.

There are two output files from the GENALTR utility:

- An ALTER dataset containing ALTER DDL – see ALTOUT sample output below.
- An FRSPC dataset containing free space information – see ALTFPC sample output below.

The output dataset’s name has the format PREFIX.USERID.ALTER.*, or it can be a user-specified name.

The summary dataset’s name has the format PREFIX.USERID.FRSPC.*.

The key to generating the ALTER statements is the difference between the NPQTY and the PQTY in the LSTCAT output. If the difference is positive, then the ALTER statement is generated.

The NSQTY value is derived as follows: if the input dataset contains a numeric value in the NSQTY field, then that value is used; if it contains the word DEFAULT, then the default secondary percentage
### Figure 1: Example input

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>OBJECT</th>
<th>PART</th>
<th>VOLSER</th>
<th>NUPGS</th>
<th>QTY</th>
<th>SQTY</th>
<th>N%use</th>
<th>NPQTY</th>
<th>NSQTY</th>
<th>NP%use</th>
<th>PART</th>
<th>OBNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXTESTDB</td>
<td>TBADETTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>1836Ø</td>
<td>288Ø</td>
<td>144Ø</td>
<td>7344Ø</td>
<td>7488Ø</td>
<td>288Ø</td>
<td>98.Ø7</td>
<td>ØØ1</td>
<td>TBADETTS TN</td>
</tr>
<tr>
<td>XXTESTDB</td>
<td>TBADBMTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>1Ø8Ø</td>
<td>288Ø</td>
<td>72Ø</td>
<td>432Ø</td>
<td>5Ø4Ø</td>
<td>144Ø</td>
<td>85.71</td>
<td>ØØ1</td>
<td>TBASUMTS TN</td>
</tr>
<tr>
<td>XXTESTDB</td>
<td>TBANOTTTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>1Ø8Ø</td>
<td>288Ø</td>
<td>72Ø</td>
<td>432Ø</td>
<td>5Ø4Ø</td>
<td>144Ø</td>
<td>85.71</td>
<td>ØØ1</td>
<td>TBANOTTTS TN</td>
</tr>
<tr>
<td>XXTESTDB</td>
<td>TBAISSTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>1Ø8Ø</td>
<td>288Ø</td>
<td>72Ø</td>
<td>432Ø</td>
<td>5Ø4Ø</td>
<td>144Ø</td>
<td>85.71</td>
<td>ØØ1</td>
<td>TBAISSTS TN</td>
</tr>
<tr>
<td>XXTESTDB</td>
<td>TBABAUTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>216Ø</td>
<td>288Ø</td>
<td>72Ø</td>
<td>864Ø</td>
<td>1152Ø</td>
<td>144Ø</td>
<td>75.ØØ</td>
<td>ØØ1</td>
<td>TBABAUTS TN</td>
</tr>
<tr>
<td>XXTESTDB</td>
<td>TBABCLTS</td>
<td>ØØ1</td>
<td>ØØ1</td>
<td>36Ø</td>
<td>288Ø</td>
<td>72Ø</td>
<td>288Ø</td>
<td>216Ø</td>
<td>144Ø</td>
<td>66.66</td>
<td>ØØ1</td>
<td>TBABCLTS TN</td>
</tr>
</tbody>
</table>
specified will be applied to the NPQTY and rounded off to the next higher cylinder boundary.

The utility also prompts the user to perform FREE SPACE analysis. This is done by calling the CHKVTOC utility, which invokes the IBM supplied IEHLIST utility with those volume names and retrieves the necessary information. It then processes the key information to get the free space availability on the volume and reports it back.

Note: the information returned about the free space availability on the volume must be analysed carefully. If the additional space being requested on a volume is 150 cylinders and if the utility indicates that the free space available is 150 cylinders or even 175 cylinders, it does not indicate a perfect fit. The additional space requirement is calculated using the new primary quantity being requested. However, when DB2 reorganizes the object, it is preferable to have the new primary quantity space available in one extent (or a maximum of five extents). If DB2 cannot get this, there could be serious problems, possibly with dataset loss.

Another use for this utility is in re-sizing a test database to hold production volumes of data. The LSTCAT utility can be run on the production databases with a one percent default increase, and then the GENALTR utility can be run on the LSTCAT output to generate the ALTER statements for all objects to reflect the correct quantity used. This way we can optimize the space requirements on the TEST database.

Other utilities to generate REORG, image copy, and RUNSTATS JCL using the output from the LSTCAT utility can be written along these lines, thereby aiding productivity.

Example input is shown in Figure 1.

ALTOUT

Sample output:

```sql
ALTER TABLESPACE XXTESTDB.TBADETTS
PRIQTY 74880 SECQTY 2880 ;
```
ALTER TABLESPACE XXTESTDB.TBADBMTS
  PRIQTY 5040 SECQTY 1440 ;
ALTER TABLESPACE XXTESTDB.TBANOTTS
  PRIQTY 5040 SECQTY 1440 ;
ALTER TABLESPACE XXTESTDB.TBAISSTS
  PRIQTY 5040 SECQTY 1440 ;
ALTER TABLESPACE XXTESTDB.TBABAUTS
  PRIQTY 11520 SECQTY 1440 ;
ALTER TABLESPACE XXTESTDB.TBABCLTS
  PRIQTY 2160 SECQTY 1440 ;

ALTFSPC

Sample output:

-------------------------
Volume    Cylreq    Cylfree
-------------------------
VOL001     121       444
-------------------------

CHKVTOC

/* REXX */
/* */
/* Invocation tso CHKVTOC VOLUMENAME */
/* */
/* The sysprint dataset is also present and the code can be */
/* turned on or off to browse the same */
TRACE o

PREFX = SYSVAR(SYSPREF)
PARSE UPPER ARG P_volnam
if strip(P_volnam)='' | strip(P_volnam) = '' then
do
  say 'Proper execution is LSTVTOC VOLUME NAME ...'
  exit(8)
end

x = outtrap("zap.","*")
CALL P1000_Allocate_Sysin
CALL P2000_Allocate_Output
Call P3000_Execute_IEHLIST
Call P4000_Clean_up
x = outtrap("OFF")

EXIT
/* */
/* */
P1000_Allocate_Sysin:
P_sysin = PREFIX||'.'||USERID()||'.SYSIN.VTOC'
address tso "delete "P_sysin"
Sysin = SYSDSN("P_sysin")
if Sysin /= "OK" then do
  address tso "ALLOCATE DDNAME(SYSIN) NEW UNIT(SYSDA) SPACE(1,1), TRACKS REUSE DSNAME('P_sysin')"
end
else do
  SAY '*** Error *** 'P_sysin Sysin
  exit(8)
end
sin.1 = " LISTVTOC FORMAT,VOL=3390="||P_volnam
  "execio * diskw SYSIN (FINIS stem sin."
RETURN
/* */
/* */
P2000_Allocate_Output:

address tso "delete '"||PREFIX||".'||USERID()||".SYSPRINT''
address tso "delete '"||PREFIX||".'||USERID()||".TEMPVT''
address tso "ALLOC DDNAME(SYSPRINT) NEW UNIT(SYSDA) space(2,2), cyl reuse DSNAME('PREFX||".'||USERID()||".SYSPRINT')"

address tso "ALLOC F(DDNAME1) NEW UNIT(3390) VOLUME("||P_volnam||"),
  "tracks SPACE(1,1) DSNAME('HRDBA."||USERID()||".TEMPVT')"

Return
/* */
/* */
P3000_Execute_IEHLIST:

  "execio * DISKR SYSPRINT (FINIS STEM prt."
lst = prt.0
linreq = last-2
out.1 = prt.linreq
say last out.1

  "execio * DISKW SYSPRINT (FINIS STEM out."

/* comment the signal code below to browse SYSPRINT dataset */
/* this may be done for debugging */
Free weekly news by e-mail

Xephon has four weekly news services covering networks, distributed systems, the data centre, and software.

Each week, subscribers receive, by e-mail, a short news bulletin consisting of a list of items; each item has a link to the page on our Web site that contains the corresponding article. Each news bulletin also carries links to the main industry news stories of the week.

To subscribe to one or more of these news services, or review recent articles, point your browser at http://www.xephon.com/newz.html.
DB2 PLAN_TABLE – access and maintenance

DB2 stores SQL access path-related details for each program in a user-supplied table called PLAN_TABLE. This table must exist for any bind process with the EXPLAIN option equal to YES. The information in the PLAN_TABLE is used in designing tables and indexes, and helps in the performance tuning of SQL queries used in application programs. A PLAN_TABLE in a production environment contains EXPLAIN results for all programs moved to the production environment, and, therefore, this data must be maintained in the same way as other production data. This article describes how accesses to a PLAN_TABLE can be made more efficiently, resulting in CPU and elapsed time savings. The article also provides some useful tips on how to maintain a PLAN_TABLE in a production environment.

EXPLAIN AND PLAN_TABLE
EXPLAIN is a monitoring tool that produces information about a plan, package, or SQL statement when it is bound. The output from EXPLAIN appears in a table called a PLAN_TABLE. The information in a PLAN_TABLE helps in determining the access path chosen for a query, designing databases, indexes, and application programs, and determining when to rebind an application.

POPULATING A PLAN_TABLE
Mostly, a PLAN_TABLE has rows inserted into it when a plan or a package is bound or rebound with the option ‘EXPLAIN (YES)’. EXPLAIN obtains information about the access paths for all explainable SQL statements in a package or the DBRMs of a plan. This information gets created in <package_owner>.PLAN_TABLE or <plan_owner>.PLAN_TABLE.

We can also populate a PLAN_TABLE by executing the SQL statement EXPLAIN and by specifying a single explainable SQL statement in the FOR clause as shown in Example 1 below. The resulting rows are created in <current_sql_id>.PLAN_TABLE.
Example 1

EXPLAIN PLAN
SET QUERYNO = 20 FOR
< explainable_select statement >

RETRIEVING ROWS FROM PLAN_TABLE
There are several processes that can insert rows in a PLAN_TABLE. In order to understand access paths, we must retrieve rows for a particular query in an appropriate order. All rows for a particular plan are identified by the value of APPLNAME or PROGNAME. All rows for a particular package are identified by the values of PROGNAME and COLLID (with no package versioning).

In order to retrieve this data for a particular package or a plan from PLAN_TABLE, a select statement like Example 2 is used – see below. Please note that the WHERE clause predicates are for PROGNAME and COLLID. If it was a package bind, we get EXPLAIN rows for that package because each package (no package versioning) can be uniquely identified by a collection-id and program name. If it was a plan bind, APPLNAME and PROGNAME will have the same value – that of the plan name – and the COLLID column is blank. Example 2 can be used to retrieve EXPLAIN rows for that particular plan.

Since the PLAN_TABLE does not have an index, any query on this table always results in a tablespace scan. If we run EXPLAIN for the SELECT statement in Example 2A, the output results (shown in Table 1) display a tablespace scan followed by an ORDERBY sort. As more and more programs are moved into production and because the same program is bound multiple times in production, this PLAN_TABLE also grows. Therefore any SELECT on this table will cost more and more CPU time as well as elapsed time. Also, if we wish to delete rows for a particular plan or package from this table, this deletion will also cause a tablespace scan, incurring a higher CPU cost.

Similarly, in order to retrieve EXPLAIN rows for a particular SQL statement that were inserted by EXPLAIN Example 1 (above) into the PLAN_TABLE, a SQL statement like Example 3 is used. The
EXPLAIN output for Example 3 (shown in Table 2) also verifies the tablespace scan followed by an ORDERBY sort.

Example 2

```sql
SELECT *
FROM SYSADM2.PLAN_TABLE
WHERE PROGNAME = 'PROG1'
    AND COLLID = 'COLLECTION1'
    AND TIMESTAMP > '1998011508000000'
ORDER BY
    QUERYNO,
    QBLOCKNO,
    PLANNO,
    MIXOPSEQ
```

Table 1

EXPLAIN results for Example 2:

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESSTYPE</th>
<th>MATCHCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>DSNESM68</td>
<td>1</td>
<td>Ø</td>
<td>SYSADM2</td>
<td>PLN_TBLE</td>
<td>R</td>
<td>Ø</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>DSNESM68</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>Ø</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACCESSCREATOR</th>
<th>ACCESSNAME</th>
<th>INDEXONLY</th>
<th>SORTC_ORDERBY</th>
<th>COLLID</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td></td>
<td>DSNESPCS</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td></td>
<td>DSNESPCS</td>
<td></td>
</tr>
</tbody>
</table>

Example 2A

```sql
SELECT *
FROM SYSADM2.PLAN_TABLE
WHERE PROGNAME = 'PROG1'
    AND COLLID = 'COLLECTION1'
    AND TIMESTAMP > '1998011508000000'
ORDER BY
    PROGNAME,
    COLLID,
    QUERYNO,
    QBLOCKNO,
    PLANNO,
    MIXOPSEQ
```

Example 3

```sql
SELECT *
FROM SYSADM2.PLAN_TABLE
```
WHERE QUERYNO = 2Ø
ORDER BY
  QBLOCKNO,
  PLANNO,
  MIXOPSEQ

Table 2
EXPLAIN results for Example 3:

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESSTYPE</th>
<th>MATCHCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Ø</td>
<td>1</td>
<td>DSNESM68</td>
<td>1</td>
<td>Ø</td>
<td>SYSADM2</td>
<td>PLN_TABLE</td>
<td>R</td>
<td>Ø</td>
</tr>
<tr>
<td>2Ø</td>
<td>1</td>
<td>DSNESM68</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td>Ø</td>
<td></td>
</tr>
</tbody>
</table>

ACCESSCREATOR ACCESSNAME INDEXONLY SORTC_ORDERBY COLLID

N        N        DSNESPCCS
N        Y        DSNESPCCS

INDEX ON PLAN_TABLE
This tablespace scan can be avoided by creating a clustering index, say PLAN_INDEX, with keys as described in the following SQL statement:

CREATE INDEX SYSADM2.PLAN_INDEX
  ON SYSADM2.PLAN_TABLE
  (PROGNAME ,
   COLLID ,
   QUERYNO ,
   QBLOCKNO ,
   PLANNO ,
   MIXOPSEQ ,
   TIMESTAMP )
  USING VCAT VCAT1
  FREEPAGE Ø PCTFREE  5
  CLUSTER

With this index, Example 2, which retrieves output rows for a particular plan/package, can now be modified to Example 2A (above) and the EXPLAIN results for statement 2A are described in Table 3 (also below). We can see that the access path has now improved to an index scan with MATCHCOLS = 2 and does not have an ORDERBY sort. This definitely results in less CPU and elapsed time as compared to the example without an index. In a PLAN_TABLE containing 190,000 rows, retrieving 7 rows for a program, use of an index (see
Figure 1) brings down the CPU time to 0.22 seconds as compared to 1.24 seconds of CPU without using an index. These output rows for a plan or package can later be deleted from the PLAN_TABLE with SQL Example 4, which involves an index scan with MATCHCOLS = 2.

If the EXPLAIN rows were created for a particular SQL statement by using the SQL statement EXPLAIN as in Example 1, then these rows can be retrieved by changing Example 3 to Example 3A in order to include PROGNAME and COLLID in the ‘where’ clause and ‘order by’ clause as shown in Example 3A. EXPLAIN results for statement 3A are shown in Table 4, depicting improvements with an index scan with MATCHCOLS = 3 and with no ORDERBY sort. In a PLAN_TABLE containing 190,000 rows, retrieving 4 rows for a query, using an index (see Figure 1) brings down CPU time to 0.19 seconds as compared to 1.17 seconds of CPU without using an index.

Please note that, here, PROGNAME and COLLID correspond to the DYNAMIC SQL program used for running the SQL statement EXPLAIN. It can be a SPUFI, QMF, or DSNTEP2, or any other program depending on which one was used to create the rows in the PLAN_TABLE. (We can also find out values for PROGNAME and COLLID as a one-time exercise by browsing through PLAN_TABLE rows for column QUERYNO equal to the value set for QUERYNO during execution of the SQL statement EXPLAIN.) In our example, we have COLLID = DSNEPCS and PROGNAME = DSNEM68. These EXPLAIN rows can later be deleted by Example 5, which involves an index scan with MATCHCOLS = 3.

In all cases, by introducing an index on a PLAN_TABLE, we can see significant improvements in CPU and elapsed times.

<table>
<thead>
<tr>
<th>Statement 2</th>
<th>Statement 2A</th>
<th>Statement 3</th>
<th>Statement 3A</th>
</tr>
</thead>
<tbody>
<tr>
<td>(without index)</td>
<td>(with index)</td>
<td>(without index)</td>
<td>(with index)</td>
</tr>
<tr>
<td>1.24</td>
<td>0.22</td>
<td>1.17</td>
<td>0.19</td>
</tr>
</tbody>
</table>

Figure 1: CPU times (in seconds)
Table 3
EXPLAIN results for Statement 2A (with an index on the PLAN_TABLE):

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESSTYPE</th>
<th>MATCHCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Ø</td>
<td>1</td>
<td>DSNESM68</td>
<td>1</td>
<td>Ø</td>
<td>SYSADM2</td>
<td>PLN_TABLE</td>
<td>I</td>
<td>2</td>
</tr>
</tbody>
</table>

ACCESSCREATOR  ACCESSNAME INDEXONLY SORTC_ORDERBY COLLID

SYSADM2        PLAN_INDEX N       N       DSNESPCS

Example 3A

SELECT *
FROM SYSADM2.PLAN_TABLE
WHERE PROGNAME = 'DSNESM68'
AND COLLID = 'DSNESPCS'
AND QUERYNO = 4Ø
ORDER BY
    PROGNAME,
    COLLID,
    QUERYNO,
    QBLOCKNO,
    PLANNO,
    MIXOPSEQ

Table 4
EXPLAIN results for Statement 3A (with an index on the PLAN_TABLE):

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESSTYPE</th>
<th>MATCHCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4Ø</td>
<td>1</td>
<td>DSNESM68</td>
<td>1</td>
<td>Ø</td>
<td>SYSADM2</td>
<td>PLN_TABLE</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

ACCESSCREATOR  ACCESSNAME INDEXONLY SORTC_ORDERBY COLLID

SYSADM2        PLAN_INDEX N       N       DSNESPCS

Example 4

DELETE
FROM SYSADM2.PLAN_TABLE
WHERE PROGNAME = 'PROG1'
    AND COLLID = 'COLLECTION1'
    AND TIMESTAMP > '1998011500000000'
Example 5

DELETE
FROM SYSADM2.PLAN_TABLE
WHERE PROGNAME = 'DSNESM68'
AND COLLID = 'DSNESPES'
AND QUERYNO = 40

MAINTAINING EXPLAIN RESULTS FOR MULTIPLE BINDS AND REBINDS

Because the same program is bound and rebound many times in production, these EXPLAIN output rows keep on accumulating in the PLAN_TABLE with different values for the timestamp. As a policy, we must maintain EXPLAIN output data for the earlier versions (at least the previous version) of each program in a PLAN_TABLE. This helps when comparing the access paths of the current version of a program with that of the earlier one, observing differences if any, analysing reasons for those differences, and taking appropriate action. This may be needed when a program performs badly after a rebind (because of a change in catalog statistics) or when a new version of a program is moved to production and it starts performing poorly.

In a PLAN_TABLE, each set of rows for a particular version of a program can be identified only by a range of timestamp values. Therefore, retrieval of these rows for the current version of a program, or for any earlier version, becomes very difficult because we have to first find out the timestamp values for the first and last row in the set before we can execute the actual SELECT statement for retrieval of rows.

PLAN HISTORY TABLE

One of the alternative ways to resolve this issue is to create a plan history table called PLAN_HIST_TABLE, which is a mirror image of the PLAN_TABLE. Since most of the time we are concerned with EXPLAIN results only for current versions of programs in production, it is a good idea to transfer all existing rows for a program from the PLAN_TABLE to a plan history table and delete those rows in the PLAN_TABLE before we perform a bind or rebind for that program.
The steps to be taken during each bind/rebind of a plan or a package are shown below:

- **Step 1:**
  
  ```sql
  INSERT INTO PLAN_HIST_TABLE
  SELECT *
  FROM PLAN_TABLE
  WHERE PROGNAME  =  'PROG1'
  AND COLLID    =  'COLLECTION1'
  ```

- **Step 2:**
  
  ```sql
  DELETE FROM PLAN_TABLE
  WHERE PROGNAME  =  'PROG1'
  AND COLLID    =  'COLLECTION1'
  ```

- **Step 3 – bind/rebind package or plan.**

  By doing this, we can maintain data for all earlier versions of a program in a plan history table and a PLAN_TABLE will contain data only for current versions of all programs. Because the plan history table will have many more rows than the PLAN_TABLE, we need to build an index (similar to PLAN_INDEX) on the PLAN_HIST_TABLE, which could be called PLAN_HIST_INDEX, and which will provide efficient access for any query made on the plan history table.

  As this history table grows with time, we must plan to delete rows for older versions of the program that may not be of any importance for the installation. We may also like to have a policy to keep up to the last two versions of a program in this table.

**PLAN TABLE AND PLAN HISTORY TABLE MAINTENANCE**

As we can see, there are frequent inserts and deletes in a PLAN_TABLE and plan history table. Therefore, these tables become candidates for periodic reorganization (the frequency of reorganization will depend on how many binds take place in a day, week, or month, and when the rows are deleted from these tables). A RUNSTATS taken at the appropriate time on these tables will ensure improved access on them.

Also, these tables contain a very important source of information that
is required for the performance tuning of programs, making database changes to improve access paths, etc. We must take image copies of these tables regularly, like any other production tables.

CONCLUSION

PLAN_TABLE data in production is a very important source of information for making changes to application programs, database designs, and performance tuning. In order to avoid any data loss, this table should be regularly image-copied. Also, this table is volatile and therefore it must be re-organized periodically in order to maintain efficient access to any query on this table.

Sharad Kumar Pande
Senior DB2 DBA
PricewaterhouseCoopers (USA)

Contributing to DB2 Update

In addition to DB2 Update, the Xephon family of Update publications now includes CICS Update, MVS Update, VSAM Update, TCP/SNA Update, RACF Update, AIX Update, Domino Update, MQ Update, NT Update, Oracle Update, SQL Server Update, and TSO/ISPF Update. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

If you have ever experienced any difficulties, or made an interesting discovery which would be of interest to our readers, you could receive a cash payment, a free subscription to any of our Updates, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it. For a copy of our Notes for Contributors, which explains the terms and conditions under which we publish articles, please contact the editor at any of the addresses shown on page 2.
This month we conclude with the code that makes it simpler to prepare image copy jobs for tablespaces and DSNTIAUL copy jobs for tables that require GDGs to be defined, and to write JCL for each object created.

```
WRITE_SYSIN:PROCEDURE EXPOSE UPPERL LOWERL TSPARTCNT. TSNAMES.
    DO MAIN_LP=UPPERL TO LOWERL BY -1
        TSNAME=STRIP(SUBSTR(TSNAMES.MAIN_LP,1,8),'B') || '.' || ,
            STRIP(SUBSTR(TSNAMES.MAIN_LP,9,8),'B')
    IF TSPARTCNT.MAIN_LP=Ø THEN DO
        LINE = '   COPY TABLESPACE ' || TSNAME || ,
            ' COPYDDN S'||MAIN_LP||' SHRLEVEL CHANGE'
            PUSH LINE
        END
    ELSE DO
        DO PARTNUM=TSPARTCNT.MAIN_LP TO 1 BY -1
            DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø',' ')||,
                TRANSLATE(FORMAT(PARTNUM,3),'Ø',' ');
            LINE = '   COPYDDN S'||DD_NAME||' SHRLEVEL CHANGE'
                PUSH LINE
            LINE = '   COPY TABLESPACE ' || TSNAME || ' DSNUM ' || PARTNUM
                PUSH LINE
        END
    END
    PUSH '//DSNUPROC.SYSIN DD *
£EXECIO * DISKW ICJCL£
RETURN;

JOB_CARD_ICOPY:PROCEDURE EXPOSE DB2ID JC JOB_NUM
    JOB_NUM=JOB_NUM+1
    PUSH ''
    LINE='//ICOPY'||JOB_NUM||' EXEC DSNUPROC,SYSTEM=DB'||DB2ID||'Ø' ,
        '.UID='ICPY'||DB2ID||JOB_NUM||',UTPROC=''''
        PUSH LINE
    PUSH '// MSGLEVEL=(1,1),REGION=ØM,NORMY=SKMXSYP,TYPRUN=HOLD'
    LINE='//ICOPY'||JOB_NUM||' JOB (ACCT#),IMAGECOPY','||,
        'MSGCLASS=X,CLASS='||JC||','''
        PUSH LINE
£EXECIO * DISKW ICJCL£
RETURN

JOB_CARD_UNLD:PROCEDURE EXPOSE DB2ID JC
    PUSH ''
    LINE='//         DD DISP=SHR,DSN='||DB2ID||'DSN.SDSNEXIT'
```
PUSH LINE
LINE='//JOBLIB   DD DISP=SHR,DSN='||DB2ID||'DSN.SDSNLOAD'
PUSH LINE
PUSH ' // MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
PUSH ' //UNLOAD   JOB (ACCT#),"UNLOAD",MSGCLASS=X,CLASS='||JC||','
EXECIO * DISKW UNJCLE
RETURN
UNLOAD_SYSIN:PROCEDURE EXPOSE TBNAME. REMAINDER UN_IN ,
DB2ID FROM_TAB TO_TAB JCLEXT OLDJCLEXT
OLDJCLEXT=JCLEXT
DO WHILE ( OLDJCLEXT = JCLEXT )
  JCLEXT=TIME(S)
END
PUSH ''
PUSH '/*'
DO MAIN_LP=FROM_TAB TO_TAB BY -1
  TBNAME=STRIP(SUBSTR(TBNAME.MAIN_LP,1,8),'B') || '.' || ,
    TRANSLATE(STRIP(SUBSTR(TBNAME.MAIN_LP,11,18),'B'),' ','ØØ'X)
  LINE = '   SELECT * FROM '||TBNAME||' WITH UR;'
PUSH LINE
END
PUSH '//'
PUSH '/*'
DO SYSIN DD *'
PUSH '//'
  DISP=(NEW,DELETE,DELETE),SPACE=(8ØØ,(29Ø,9Ø),RLSE)
LINE='//SYSPUNCH DD DSN=SYSPDBA.PSØ.UNLOAD.JCL'||JCLEXT||'.TEMP,'
PUSH LINE
EXECIO * DISKW UNJCLE
RETURN
NEW_STEP:PROCEDURE EXPOSE INT_PART DB2ID
PUSH '
PUSH '//'
SYSSDUMP DD SYSOUT=*'
PUSH '//'
SYSSPRINT DD SYSOUT=*'
LINE='    LIB('''||DB2ID||'DSN.RUNLIB.LOAD''))'
PUSH LINE
PUSH ' ' RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) PARMS(''SQL'') - '
LINE=' DSN SYSTEM(DB'||DB2ID||'Ø)' P
PUSH LINE
PUSH '//'
SYSTSIN DD *'
PUSH '//'
SYSTSPRT DD SYSOUT=*'
LINE='//UNLD'||INT_PART||' EXEC PGM=IKJEFTØ1,REGION=ØM'
PUSH LINE
EXECIO * DISKW UNJCLE
RETURN
ADD_DD_COPY:PROCEDURE EXPOSE MAIN_LP TSNAME DB2ID ,
RETPD_COPY PARTNUM PREV_DD LABEL_NUM TSPARTCNT.
LABEL_NUM=LABEL_NUM+1;
IF TSPARTCNT.MAIN_LP > Ø THEN DO
  DSNAME='ODM.GBØ.S1D.'||TSNAME||'.P'||PARTNUM||'+1'
  DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø','' ) || ,
    TRANSLATE(FORMAT(PARTNUM,3),'Ø','' ) ;
ELSE DO
  DSNAME='ODM.GBØ.S1D.'||TSNAME||'(+1)'
  DD_NAME=TRANSLATE(FORMAT(MAIN_LP,4),'Ø',' ');
END;

PUSH ''
IF MAIN_LP = 1 & PARTNUM=1 | LABEL_NUM=1 THEN DO
  LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_COPY
  PUSH LINE
  PUSH '// VOL=(,,,2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'.
  LINE='//S'||DD_NAME||' DD DSN='||DSNAME||','
  PUSH LINE
END ELSE DO
LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_COPY
PUSH LINE
LINE='// VOL=(,RETAIN,,2Ø,REF=*.S'||PREV_DD||'),'
PUSH LINE
LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=S'||PREV_DD||',LABEL=('||LABEL_NUM||',SL),'
PUSH LINE
LINE='//S'||DD_NAME||' DD DSN='||DSNAME||','
PUSH LINE
END
PREV_DD=DD_NAME;

ÆEXECIO * DISKW ICJCLÆ
RETURN
ADD_DD_UNLD:PROCEDURE EXPOSE INT_PART REMAINDER TBNAME ,
    LAST_DD_NAME MAIN_LP UNL_DS_NAME UN_IN DB2ID RETPD_UNLD
  DD_ID=REMAINDER
  IF REMAINDER = Ø THEN DD_ID=UN_IN
  CURRENT_PTR=TRANSLATE(FORMAT(DD_ID - 1,2),'Ø',' ')
  PREV_PTR=TRANSLATE(FORMAT(DD_ID - 2,2),'Ø',' ')
  PUSH ''
  IF DD_ID = 1 THEN DO
    LINE='// DCB=BLKSIZE=1Ø24,TRTCH=COMP,BUFNO=2Ø,RETPD='||RETPD_UNLD
    PUSH LINE
    IF INT_PART=Ø THEN ,
    PUSHEH REPLACE=*,LASTDD),'
    PUSH LINE
    LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=LASTDD'|| ,
      ',LABEL=('||MAIN_LP||',SL),'
    PUSH LINE
    LINE='//SYSRECØØ DD DSN=UNLOAD.GBØ.'|| .
    UNL_DS_NAME||'(+1)'||'.','
  ELSE DO
    LINE='// VOL=(,,,2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'
    ELSE DO
      LINE='// VOL=(,RETAIN,,2Ø,REF=*.LASTDD),'
      PUSH LINE
      LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=LASTDD'|| ,
        ',LABEL=('||MAIN_LP||',SL),'
      PUSH LINE
    END
    LINE='//SYSRECØØ DD DSN=UNLOAD.GBØ.'|| .
    UNL_DS_NAME||'(+1)'||'.','
  ELSE DO
    LINE='// VOL=(,...2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'
    ELSE DO
      LINE='// VOL=(,RETAIN,,2Ø,REF=*.LASTDD),'
      PUSH LINE
      LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=LASTDD'|| ,
        ',LABEL=('||MAIN_LP||',SL),'
      PUSH LINE
    END
    LINE='//SYSRECØØ DD DSN=UNLOAD.GBØ.'|| .
    UNL_DS_NAME||'(+1)'||'.','
  ELSE DO
    LINE='// VOL=(,...2Ø),DISP=(NEW,CATLG,CATLG),UNIT=CARTM,LABEL=(1,SL),'
  ELSE DO
    LINE='// VOL=(,RETAIN,,2Ø,REF=*.LASTDD),'
    PUSH LINE
    LINE = '// DISP=(NEW,CATLG,CATLG),UNIT=AFF=LASTDD'|| ,
      ',LABEL=('||MAIN_LP||',SL),'
    PUSH LINE
  END
PUSH LINE
IF INT_PART<>Ø THEN DO
  LINE='//LASTDD   DD DSN='||LAST_DD_NAME||',DISP=OLD'
  PUSH LINE
END
END
ELSE DO
  LINE='// VOL=(.RETAIN,.20,REF=*SYSREC'||PREV_PTR||')'
  PUSH LINE
LINE='// DCB=BLKSIZE=3276Ø,TRTCH=COMP,BUFIN=2Ø,RETPD='||RETPD_UNLD||',
  LABEL=('||MAIN_LP||'.SL),'
  PUSH LINE
LINE='//SYSREC'||CURRENT_PTR||' DD DSN=UNLOAD.GBØ.'||,
  UNL_DS_NAME||(1)||',
  PUSH LINE
LAST_DD_NAME='UNLOAD.GBØ.'||,
  UNL_DS_NAME||(1);
END
£EXECIO * DISKW UNJCL£
RETURN
DO_SELECT:PROCEDURE EXPOSE SEL. DB2ID
  PUSH ''
  DO X = SEL.Ø TO 1 BY -1
    PUSH SEL.X
  END
£EXECIO * DISKW SYSIN (FINISE£
CMD = £RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL)£
CMD = CMD || £LIB('||DB2ID||'DSN.RUNLIB.LOAD')£
CMD = CMD || £PARMS('SQL')£
QUEUE 'END'
IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
IF RC > Ø THEN DO
  SAY 'CAN NOT CONNECT TO DB2 SUBSYSTEM.'
  SAY 'PLEASE TRY LATER...'
  RETURN
END
QUEUE CMD
QUEUE 'END'
IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
£EXECIO * DISKR SYSPRINT (STEM SQLHATA.£
UNLD_OK=Ø
DO SQ_LP=1 TO SQLHATA.Ø
   IF INDEX(SQLHATA.SQ_LP,'DSNT495I SUCCESSFUL UNLOAD') > Ø THEN,
      UNLD_OK=1
   END
IF UNLD_OK=Ø THEN DO
   SAY 'THERE IS AN ERROR IN SQL STATEMENT.'
   DO SQ_LP2=1 TO SQLHATA.Ø
      SAY SQLHATA.SQ_LP2
   END
   EXIT 2Ø
END
PUSH ''
£EXECIO * DISKW SYSPRINT (FINIS£
RETURN
ADD_WTO:PROCEDURE EXPOSE OPR_TYPE
PUSH ''
PUSH '//'
PUSH '/*'
IF OPR_TYPE = 'UNLOAD' THEN DO
   PUSH '->****************************************************'
PUSH '->SYS2.BACKUP.JCLLIB(UNLRES) JCL.'
PUSH '->SYS2.OPERLIB(UNLRESTB) AND RESTART THE JOB WITH '
PUSH '->TABLE NAME DUMPED SUCCESSFULLY TO THE DATASET '
PUSH '->UNLOAD JOB ENDED WITH ERROR. PLEASE WRITE THE LAST '
PUSH '->****************************************************'
END
ELSE DO
   PUSH '-> ****************************************************'
PUSH '-> ARE COMPLETED.'
PUSH '-> SYS2.BACKUP.JCLLIB(ICOPYX) AFTER ALL ICOPYX JOBS'
PUSH '-> TABLESPACE, RESTART THE JOB WITH THE JCL '
PUSH '-> ERROR. WHEN YOU ARE SURE THAT THERE IS NO RESTRICTED'
PUSH '-> ICOPY JOB ENDED WITH ERROR. PLEASE INVESTIGATE THE '
PUSH '-> ****************************************************'
END
PUSH '/*/SYSIN DD *'
PUSH '*/ERROR EXEC IPOWTO,REGION=768K,COND=((4,GE),EVEN)'
IF OPR_TYPE = 'UNLOAD' THEN £EXECIO * DISKW UNJCL£
IF OPR_TYPE = 'COPY' THEN £EXECIO * DISKW ICJCL£
RETURN

ICOPYX

//ICOPYX JOB (ACCT#),''(MSGCLASS=X,CLASS=9,
// MSGLEVEL=(1,1),REGION=4M
//***************************************************************
//* ICOPYALL DB2ID OP_ID UNLINT COPY_JCL_NAME UNLOAD_JCL_NAME
*/ RETPD_COPY RETPD_UNLD TS_SELECT TB_SELECT PARTITION JOB_CLASS */

DB2ID : DB2 SUBSYSTEM ID. IT MAY BE D, T, E, OR P */

OP_ID : MAY BE UNLD, COPY OR BOTH. */

UNLD : ONLY DSNTIAUL COPY. */

COPY : ONLY IMAGE COPY. */

BOTH : BOTH OF DSNTIAUL COPY AND IMAGE COPY. */

UNLINT : CREATES A NEW STEP EVERY UNLINT DD. */

COPY_JCL_NAME : IMAGE COPY JCL TO BE CREATED. */

UNLOAD_JCL_NAME : DSNTIAUL JCL TO BE CREATED. */

RETPD_COPY : RETENTION PERIOD OF IMAGE COPY DATASETS. */

RETPD_UNLD : RETENTION PERIOD OF DSNTIAUL COPY DATASETS. */

TS_SELECT : QUERY THAT SELECTS TABLESPACES TO BE IMAGE COPIED. */

TB_SELECT : QUERY THAT SELECTS TABLES TO BE COPIED USING DSNTIAUL. */

JOB_CLASS : JOB CLASS OF THE JCLS. */

JOB_COUNT : JOB COUNT. */

PARTITIONED : 'YES' IF IMAGE COPIES WILL BE TAKEN PARTITIONED TABLESPACE LEVEL, OTHERWISE 'NO'. */

IS_RESTART : MUST BE CODED 'RESTART', IF DSNTIAUL COPY IS TO BE RESTARTED. */

RECALL */

REXX */

PARSE ARG DB2ID OP_ID RECTS_DSNAM REC_IXDSNAME JOB_COUNT PARTITION STEP_CNT=0
CNT=0
G_STEP_CNT=0
JOB_CNT=1

MAX KILOBYTES AFTER THAT WE CATALOG WORK SPACES */

MAX_KB_WORK_CATLG=40000;
/* MAX KILOBYTES THAT A DASD CAN ALLOCATE */
MAX_KB_WORK=2600000
TARIH = DATE('E')
SAAT = TIME()
$ALLOC FI(SYSPRINT) RECFM(F B) LRECL(133) SPACE(1,1) BLOCK(4096)$
$ALLOC FI(SYSPRNCH) RECFM(F B) LRECL(80) SPACE(1,1) BLOCK(4096)$
$ALLOC FI(SYSIN) RECFM(F B) LRECL(80) BLKSIZE(80)$
$ALLOC FI(RETSJCL) DA('SYS2.BACKUP.JCLLIB('||RECTS_DSNAME||')') SHR$
$ALLOC FI(LISTCATO) RECFM(V B) LRECL(125) SPACE(1,1) BLOCK(629)$
CALL JOB_CARD_RECOVER_TS
SEL.Ø=9
SEL.1=' SELECT DBNAME,NAME'
SEL.2=' FROM SYSIBM.SYSTABLESPACE '
SEL.3=' WHERE DBNAME NOT LIKE ''WRK%'' AND NAME NOT LIKE ''UNLOAD%''
SEL.4=' AND NAME NOT LIKE ''CPY%''
SEL.5=' AND DBNAME <> 'BMCARM'
SEL.6=' AND DBNAME NOT LIKE 'DSN%'' DSQ%''
SEL.7=' ORDER BY DBNAME,NAME:'
$ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4096)$
CALL DO_SELECT
$EXECIO * DISKR SYSRECØØ (STEM TSNames.$
$EXECIO * DISKR SYSRECØØ (FINIS$
$FREE FI(SYSRECØØ)$
REC_CNT=Ø;
DO MAIN_LP=1 TO TSNames.Ø
SAY MAIN_LP
DBNAME=STRIP(SUBSTR(TSNames.MAIN_LP,1,8),'B')
TSNAME=STRIP(SUBSTR(TSNames.MAIN_LP,9,8),'B')
IF PARTITION='YES' THEN DO
SEL.Ø=3;
SEL.1=' SELECT CHAR(DECIMAL(PARTITIONS)) ' FROM SYSIBM.SYSTABLESPACE '
SEL.2=' WHERE NAME='TSNAME''
$ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4096)$
CALL DO_SELECT
$EXECIO * DISKR SYSRECØØ (STEM PARTCnt.$
$EXECIO * DISKR SYSRECØØ (FINIS$
$FREE FI(SYSRECØØ)$
PARTCnt.1=SUBSTR(PARTCnt.1,2,5); PARTCnt.1=STRIP(TRANSLATE(PARTCnt.1,' ','ØØ'X),'B'); DO WHILE ( SUBSTR(PARTCnt.1,1,1) = 'Ø' & LENGTH(PARTCnt.1) > 1 )
IF SUBSTR(PARTCnt.1,1,1)='Ø' THEN ,
   PARTCnt.1=SUBSTR(PARTCnt.1,2,LENGTH(PARTCnt.1)-1)
END;
TSPARTCnt=PARTCnt.1 END ELSE TSPARTCnt=Ø
IF TSPARTCNT=Ø THEN PARTNUM_LAST=1
ELSE PARTNUM_LAST=TSPARTCNT
DO PARTNUM=1 TO PARTNUM_LAST
   REC_CNT=REC_CNT+1;
   VOLCNT.REC_CNT=Ø;
   IF TSPARTCNT=Ø THEN NUMPART.REC_CNT=Ø
   ELSE NUMPART.REC_CNT=PARTNUM
   DBNAME.REC_CNT=DBNAME
   TSNAME.REC_CNT=TSNAME
   SEL.Ø=7;
   SEL.1=' SELECT DSNAME,TIMESTAMP FROM SYIBM.SYSCOPY'
   SEL.2=' WHERE DBNAME='||''''||DBNAME||''''||' AND'
   SEL.3='       TSNAME='||''''||TSNAME||''''||' AND'
   SEL.4='       DSNAME LIKE '||''''||'ODM%'||''''||' AND'
   IF TSPARTCNT=Ø THEN ,
      SEL.5='       DSNUM=Ø AND'
   ELSE ,
      SEL.5='       DSNUM='||PARTNUM||' AND'
   SEL.6='      ICTYPE='||''''||'F'||''''
   SEL.7=' ORDER BY TIMESTAMP DESC;
   £ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
   CALL DO_SELECT
   £EXECIO * DISKR SYSRECØØ (STEM DSNAME.£
   £EXECIO * DISKR SYSRECØØ (FINIS£
   £FREE FI(SYSRECØØ)£
   LAST_COPY_DSN.REC_CNT=STRIP(SUBSTR(DSNAMES.1,1,44),'B');
   PROFILE NOPREFIX
   £LISTCAT ENT(LAST_COPY_DSN.REC_CNT) OFILE(LISTCATO) ALLE
   IF RC <> Ø THEN DO
      SAY 'AN ERROR IS ENCOUNTERED WHILE TAKING LISTCAT...' EXIT 20
   END;
   £EXECIO * DISKR LISTCATO (STEM LISTC. £
   DO LC=1 TO LISTC.Ø
      IF INDEX(LISTC.LC,'VOLSER') > Ø THEN DO;
         VOLCNT.REC_CNT=VOLCNT.REC_CNT+1
         VOLSER.REC_CNT=SUBSTR(LISTC.LC,27,6)
      END;
   END;
   £EXECIO * DISKR LISTCATO (FINIS £
   SAY 'DB NAME=' DBNAME.REC_CNT 'TS NAME=' TSNAME.REC_CNT ,
       'VOLSER=' VOLSER.REC_CNT
   SAY 'LAST IC DSN NAME=' LAST_COPY_DSN.REC_CNT
   SAY '----------------------------------------------------'
   END
END
/* WRITE JCL */
PREV_VOLSER=''
PREV_MAIN_LP=1
DO MAIN_LP=1 TO REC_CNT
IF PREV_VOLSER <> VOLSER.MAIN_LP & PREV_VOLSER <> '' | ,
   VOLCNT.MAIN_LP > 1 THEN DO;
   FIRST_CNT=PREV_MAIN_LP
   LAST_CNT=MAIN_LP-1
   CALL WRITE_SYSIN
   CALL JOB_CARD_RECOVER_TS
   PREV_MAIN_LP=MAIN_LP
END;
FMLP=FORMAT(MAIN_LP,6)
FMLP=TRANSLATE(FMLP,'Ø',' ')
PUSH ''
LINE='//             DISP=(OLD, PASS)'
PUSH LINE
IF MAIN_LP = PREV_MAIN_LP THEN ,
   LINE='//             UNIT=(,.DEFER),'
ELSE ,
   LINE='//             UNIT=AFF=DD'||OLD_FMLP||',,'
PUSH LINE
LINE='//             VOL=(,RETAIN),'
PUSH LINE
LINE='//DD'||FMLP||' DD DSN='||LAST_COPY_DSN.MAIN_LP||','
PUSH LINE
OLD_FMLP=FMLP
EXECIO * DISKW RETSJCL$
IF VOLCNT.MAIN_LP=1 THEN PREV_VOLSER=VOLSER.MAIN_LP
END
FIRST_CNT=PREV_MAIN_LP
LAST_CNT=MAIN_LP-1
CALL WRITE_SYSIN
PUSH ''
EXECIO * DISKW RETSJCL (FINISH
FREE FI(RETSJCL)
/*   INDEX RECOVER JCLS   */
ALLOC FI(REIXJCL) DA('SYS2.BACKUP.JCLLIB($|REC_IXDSNAME||$)') SHRE
CALL JOB_CARD_RECOVER_IX
SEL.$=25;
SEL.1=' SELECT IXCREATOR,IXNAME,,'
SEL.2=' CHAR(DECIMAL((CARDF/1024)*(TOTLEN+13),15)) FROM ('
SEL.3=' SELECT B.CREATOR AS IXCREATOR,'
SEL.4=' B.NAME AS IXNAME,A.CARDF AS CARDF,'
SEL.5=' SUM(LENGTH) AS TOTLEN'
SEL.6=' FROM SYSIBM.SYSTABLES A,SYSIBM.SYSINDEXES B,'
SEL.7=' SYSIBM.SYSKEYS C,SYSIBM.SYSCOLUMNS D,'
SEL.8=' SYSIBM.SYSTABLESPACE E'
SEL.9=' WHERE A.TSNAME=E.NAME AND '
SEL.10=' A.NAME=B.TBNAME AND'
SEL.11=' A.CREATOR=B.TBCREATOR AND'
SEL.12='         A.DBNAME=E.DBNAME AND'
SEL.13='        B.NAME=C.IXNAME AND'
SEL.14='        C.COLNAME=D.NAME AND'
SEL.15='        A.NAME=D.TBNAME AND'
SEL.16='        A.CREATOR=D.TBCREATOR AND'
SEL.17='        B.CREATOR NOT LIKE '||''''||'SYSIBM'||''''|| ' AND '
SEL.18='        B.CREATOR NOT LIKE '||''''||'UNL'||''''|| ' AND '
SEL.19='        B.CREATOR NOT LIKE '||''''||'DSN'||''''|| ' AND '
SEL.20='        E.DBNAME NOT LIKE '||''''||'DSN'||''''|| ' AND '
SEL.21='        E.DBNAME NOT LIKE '||''''||'WRK'||''''|| ' AND '
SEL.22='        E.NAME <> '||''''||'PCPYTAB'||''''|| ' AND '
SEL.23='        E.NAME NOT LIKE '||''''||'UNLOAD'||''''|| ' AND '
SEL.24='        E.NAME NOT LIKE '||''''||'DSN'||''''|| ' AND '
SEL.25='  GROUP BY B.CREATOR,B.NAME,A.CARDF) AS QRY1;'}

£ALLOC FI(SYSRECØØ) SPACE(1,1) BLOCK(4Ø96)£
CALL DO_SELECT
£EXECIO * DISKR SYSRECØØ (STEM SPACES.£
£EXECIO * DISKR SYSRECØØ (FINIS£
£FREE FI(SYSRECØØ)£
HOWMANY_IX_PER_JOB=SPACES.Ø % JOB_COUNT + 1
DO MAIN_LP=1 TO SPACES.Ø
   IX_CREATOR=SUBSTR(SPACES.MAIN_LP,1,8)
   IX_NAME=SUBSTR(SPACES.MAIN_LP,11,18)
   SPC_QTY=SUBSTR(SPACES.MAIN_LP,30,15)
   IX_CREATOR=STRIP(TRANSLATE(IX_CREATOR,' ','ØØ'X),'B')
   IX_NAME=STRIP(TRANSLATE(IX_NAME,' ','ØØ'X),'B')
   SPC_QTY=STRIP(TRANSLATE(SPC_QTY,' ','ØØ'X),'B')
   /* IF STATISTICS HAS NOT BEEN COLLECTED, SUPPOSE IT IS 1ØØØ */
   IF SUBSTR(SPC_QTY,1,1)='-' THEN DO
      SAY 'STATISTICS HAS NOT BEEN COLLECTED FOR INDEX:' IX_NAME
      SPACES.1=SUBSTR(SPC_QTY,2,LENGTH(SPC_QTY)-1)
      FAKTOR=1ØØØ
      END
   ELSE FAKTOR=1
   DO WHILE ( SUBSTR(SPC_QTY,1,1) = 'Ø' & LENGTH(SPC_QTY) > 2 )
      IF SUBSTR(SPC_QTY,1,1)='Ø' THEN ,
      SPC_QTY=SUBSTR(SPC_QTY,2,LENGTH(SPC_QTY)-1)
   END;
   SPACE_FOR_IX_REC=SPC_QTY
   SPACE_FOR_IX_REC=SPACE_FOR_IX_REC*FAKTOR
   SAY 'INDEX NAME=' IX_CREATOR '.' IX_NAME 'SPACE=' SPACE_FOR_IX_REC
   SPACE_FOR_IX_REC1=SPACE_FOR_IX_REC % 5 +1
   IF SPACE_FOR_IX_REC1 > MAX_KB_WORK THEN ,
      SPACE_FOR_IX_REC1 = MAX_KB_WORK
   SPACE_FOR_IX_REC2=SPACE_FOR_IX_REC % 2Ø +1
   CALL JOB_STEP_RECOVER_IX
   LINE = '   RECOVER INDEX ('||IX_CREATOR||'.'||IX_NAME||')'
   PUSH LINE
   £EXECIO * DISKW REIXJCLE£
PUSH '''
PUSH '/*'
EXECIO * DISKW REIXJCL$
IF SPACE_FOR_IX_RECI > MAX_KB_WORK_CATLG THEN DO
   STEP_CNT=STEP_CNT+1
   G_STEP_CNT=G_STEP_CNT+1
PUSH ''
PUSH '/*'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.SUT1.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK1.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK2.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK3.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK4.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK5.TEMP'
PUSH '     DELETE SYSPDBA.PSØ.RECI'||JOB_CNT||'.WORK6.TEMP'
PUSH '///SYSIN   DD *'
PUSH '///SYSPRINT DD SYSOUT=*'
LINE='//DELIX'||STEP_CNT||' EXEC PGM=IDCAMS,COND=(4,GE,RECIX'||,
   STEP_CNT||')'
PUSH LINE
EXECIO * DISKW REIXJCL$
END
END
PUSH ''
PUSH '/*'
EXECIO * DISKW REIXJCL (FINIS$
FREE FI(REIXJCL)$$
STAT=MSG('OFF')$$
FREE FI(SYSRECØØ)$
FREE FI(SYSPRINT)$$
FREE FI(SYSPUNCH)$$
FREE FI(SYSIN)$$
FREE FI(TBSEL)$$
FREE FI(UNJCL)$$
FREE FI(LISTCATO)$$
STAT=MSG('ON')$$
EXIT
JOB_CARD_RECOVER_TS:PROCEDURE EXPOSE DB2ID CNT TARIH SAAT
   CNT=CNT+1;
PUSH '''
PUSH '/// MSGLEVEL=(1,1),REGION=ØM,NOTIFY=SKMXYSY,TYPRUN=HOLD'
PUSH '///RECTS'||CNT||' JOB (ACCT#),''REXTS',MSGCLASS=X,CLASS=9,'$
EXECIO * DISKW RETSJCLE$
PUSH ''
PUSH '/*       CREATED ON '||TARIH||'   '||SAAT$
EXECIO * DISKW RETSJCLE$
CALL ADD_WTO;
PUSH '''
LINE='//RECOVER EXEC DSNUPROC,SYSTEM=DB'||DB2ID'||'Ø'||,'
PUSH LINE
EXECIO * DISKW RETSJCL
RETURN

JOB_STEP_RECOVER_IX:PROCEDURE EXPOSE DB2ID SPACE_FOR_IX_REC1,
SPACE_FOR_IX_REC2,
STEP_CNT G_STEP_CNT JOB_COUNT JOB_CNT,
HOWMANY_IX_PER_JOB MAX_KB_WORK_CATLG,
TARIH SAAT

STEP_CNT=STEP_CNT+1
G_STEP_CNT=G_STEP_CNT+1
INT_PART = G_STEP_CNT % HOWMANY_IX_PER_JOB
IF G_STEP_CNT=INT_PART*HOWMANY_IX_PER_JOB |,
   SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG |,
   STEP_CNT > 200 THEN DO;
   JOB_CNT=JOB_CNT+1
   STEP_CNT=1
   CALL JOB_CARD_RECOVER_IX
END

PUSH ''
PUSH '//SYSIN DD *'
PUSH '//SYSPRINT DD SYSOUT=*'
PUSH '//UTPRINT DD SYSOUT=*'
LINE='//            SPACE=(1024,('||SPACE_FOR_IX_REC1||,
    ','||SPACE_FOR_IX_REC1||'),,,ROUND),VOL=(,,,20)' PUSH LINE
IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN,
   LINE='//SYSUT1 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,'||JOB_CNT||,
       '.SUT1.TEMP,DISP=(NEW,CATLG,CATLG)' ELSE,
   LINE='//SYSUT1 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
       '.SUT1.TEMP,DISP=(NEW,DELETE,CATLG)' PUSH LINE
IF SPACE_FOR_IX_REC1 > MAX_KB_WORK_CATLG THEN DO
   LINE='//            SPACE=(1024,('||SPACE_FOR_IX_REC1||,
    ','||SPACE_FOR_IX_REC1||'),,,ROUND)' PUSH LINE
   LINE='//            DISP=(NEW,CATLG,CATLG)' PUSH LINE
   LINE='//SORTWKØ6 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
       '.WORK6.TEMP,' PUSH LINE
   LINE='//            SPACE=(1024,('||SPACE_FOR_IX_REC1||,
    ','||SPACE_FOR_IX_REC1||'),,,ROUND)' PUSH LINE
   LINE='//            DISP=(NEW,CATLG,CATLG)' PUSH LINE
   LINE='//SORTWKØ5 DD DSN=SYSPDBA.PSØ.RECI'||JOB_CNT||,
       '.WORK5.TEMP,'
PUSH LINE
LINE='//' SPACE=(1024,'|SPACE_FOR.IndexOf.REC1|',
     ',|SPACE_FOR.IndexOf.REC1|')...ROUND'
PUSH LINE
LINE='//' DISP=(NEW,CATLG,CATLG),'
PUSH LINE
LINE='//SORTWK04 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
     '.WORK4.TEMP,'
PUSH LINE
LINE='//' SPACE=(1024,'|SPACE_FOR.IndexOf.REC1|',
     ',|SPACE_FOR.IndexOf.REC1|')...ROUND'
PUSH LINE
LINE='//' DISP=(NEW,CATLG,CATLG),'
PUSH LINE
LINE='//SORTWK03 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
     '.WORK3.TEMP,'

END
LINE='//' SPACE=(1024,'|SPACE_FOR.IndexOf.REC1|',
     ',|SPACE_FOR.IndexOf.REC1|')...ROUND'
PUSH LINE
IF SPACE_FOR.IndexOf.REC1 > MAX_KB_WORK_CATLG THEN DO
  LINE='//' DISP=(NEW,CATLG,CATLG),'
PUSH LINE
  LINE='//SORTWK02 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
      '.WORK2.TEMP,'
PUSH LINE
END
ELSE DO
  LINE='//' DISP=(NEW,DELETE,CATLG),'
PUSH LINE
  LINE='//SORTWK02 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
      '.WORK2.TEMP,'
PUSH LINE
END
LINE='//' SPACE=(1024,'|SPACE_FOR.IndexOf.REC1|',
     ',|SPACE_FOR.IndexOf.REC1|')...ROUND'
PUSH LINE
IF SPACE_FOR.IndexOf.REC1 > MAX_KB_WORK_CATLG THEN DO
  LINE='//' DISP=(NEW,CATLG,CATLG),'
PUSH LINE
  LINE='//SORTWK01 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
      '.WORK1.TEMP,'
PUSH LINE
END
ELSE DO
  LINE='//' DISP=(NEW,DELETE,CATLG),'
PUSH LINE
  LINE='//SORTWK01 DD DSN=SYSPDBA.PS0.RECI'||JOB_CNT||,
`.WORK1.TEMP.'
PUSH LINE
END
LINE='//STEPLIB DD DSN='||DB2ID||'DSN.SDSNLOAD,DISP=SHR'
PUSH LINE
LINE='//RECIX'||STEP_CNT||' EXEC PGM=DSNUTILB,REGION=OM.'||,
    'PARM=''||''||'DB'||DB2ID||'0'||'.RECIX'||JOB_CNT''''
PUSH LINE
£EXECIO * DISKW REIXJCL£
RETURN
JOB_CARD_RECOVER_IX:PROCEDURE EXPOSE DB2ID JOB_CNT TARIH SAAT
PUSH ''
     /*     CREATED ON '||TARIH||'   '||SAAT
PUSH '    MSGLEVEL=(1,1),REGION=OM,NOTIFY=SKMXSYP,TYPRUN=HOLD'
LINE='//RECJIX'||  JOB_CNT ||,
    ' JOB (ACCT#),'RECIX',MSGCLASS=X,CLASS=9,'
PUSH LINE
£EXECIO * DISKW REIXJCL£
RETURN
DO_SELECT:PROCEDURE EXPOSE SEL. DB2ID
PUSH ''
DO X = SEL.Ø TO 1 BY -1
    PUSH SEL.X
END
£EXECIO * DISKW SYSIN (FINIS£
CMD = £RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUL)£
CMD = CMD || ' LIB('''||DB2ID||'DSN.RUNLIB.LOAD')'
CMD = CMD || £ PARMS('SQL')£
QUEUE 'END '
IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
IF DB2ID='G' THEN 'DSN SYSTEM(DBGØ)'
IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
IF RC > 0 THEN DO
    SAY 'CAN NOT CONNECT TO DB2 SUBSYSTEM.'
    SAY 'PLEASE TRY LATER...'
RETURN
END
QUEUE CMD
QUEUE 'END '
IF DB2ID='D' THEN 'DSN SYSTEM(DBDØ)'
IF DB2ID='T' THEN 'DSN SYSTEM(DBTØ)'
IF DB2ID='E' THEN 'DSN SYSTEM(DBEØ)'
IF DB2ID='G' THEN 'DSN SYSTEM(DBGØ)'
IF DB2ID='P' THEN 'DSN SYSTEM(DBPØ)'
£EXECIO * DISKR SYSPRINT (STEM SQLHATA.£
UNLD_OK=Ø
DO SQ_LP=1 TO SQLHATA.Ø
IF INDEX(SQ_LP,'DSNT495I SUCCESSFUL UNLOAD') > Ø THEN ,
    UNLD_OK=1
END
IF UNLD_OK=Ø THEN DO
    SAY 'THERE IS AN ERROR IN SQL STATEMENT.'
    DO SQ_LP2=1 TO SQ_LP
        SAY SQ_LP2
    END
    EXIT 2Ø
END
PUSH ''
EXECIO * DISK W SYSPRINT (FINIS$
RETURN
ADD_WTO:PROCEDURE
PUSH ''
PUSH '/*'
PUSH '->******************************************************'
PUSH '-> DO NOT SUBMIT IT AT THE LOCAL SITE.'
PUSH '-> THIS JCL IS PREPARED FOR DISASTER RECOVERY PURPOSES.'
PUSH '->******************************************************'
PUSH '/SYSIN DD *
    PUSH '/ERROR EXEC IPOWTO,REGION=ØM,COND=((4,LE),EVEN)'
EXECIO * DISK W RETSJC
RETURN
WRITE_SYSIN:
    PUSH ''
PUSH '/DSNUPROC.SYSIN DD *'
EXECIO * DISK W RETSJC
    DO LP1=FIRST_CNT TO LAST_CNT
        IF NUMPART.LP1 = Ø THEN DO
            PUSH ''
                IF OP_ID='TOCOPY' THEN DO
                    LINE = '   TOCOPY '||LAST_COPY_DSN.LP1
                    PUSH LINE
                END
                LINE = '   RECOVER TABLESPACE ' || DBNAME.LP1 || '.
                    '||TSNAME.LP1
                PUSH LINE
            EXECIO * DISK W RETSJC
        END;
        ELSE DO;
            PUSH ''
                IF OP_ID='TOCOPY' THEN ,
                    LINE = '   DSNUM ||NUMPART.LP1|| TOCOPY '||.
                        LAST_COPY_DSN.LP1
                ELSE LINE = '   DSNUM ||NUMPART.LP1
                PUSH LINE
                LINE = '   RECOVER TABLESPACE ' || DBNAME.LP1 || '.
                    '||TSNAME.LP1
        END;
    END;
Many subscribers reading *DB2 Update* will have met similar problems and come up with quite different solutions. We’d like to hear what your alternative solution is. Contact the editor, Trevor Eddolls, at any of the addresses shown on page 2 for a copy of our *Notes for Contributors*. 

---

Abdullah Ongul

DBA

Disbank (Turkey) © Xephon 2000
Embarcadero Europe has announced the availability of its ERJStudio 4.0, an upgrade to its modelling environment that offers advanced parser-based support for stored procedures and triggers and an automation interface for user customization.

Users can create stored procedures and triggers in their native DBMS languages (including PL/SQL and Transact SQL) or they can create template versions for drag-and-drop reuse. ERJStudio 4.0 ensures consistent stored procedure and trigger object dependency for tables, views, and other procedural logic.

ERJStudio has an automation interface and, by creating Sax BASIC commands (a VBA-like language) in ER/Studio’s new macro scripting UI, users can tap directly into ER/Studio’s own object model to expand existing functionality and create new functionality.

The automation interface can be used to create operations internal to ER/Studio, such as automating repetitive tasks, or between ER/Studio and virtually any other database or application environment with an exposed API or similar automation interface.

ERJStudio 4.0 runs on Windows 95, 98, NT, and 2000. Notational and method support includes IDEF1X, James Martin’s IE, and Filtered IE (designed to hide foreign keys). Supported databases include DB2 and DB2 Universal Database, as well as Oracle 7.3, 8, and 8i, Sybase 11.x, Informix SE and Online, Microsoft SQL Server 6.5 and 7.0, SQL Anywhere 5, Watcom 4, InterBase 4.x, Access 2.0, 95, 97, and 2000, and Visual FoxPro.

For further information contact:
Embarcadero, 400 Montgomery St #300, San Francisco, CA 94104, USA.
Tel: (415) 834 3131.

* * *

IBM has announced DB2 UDB Version 7, which it describes as being “designed from the ground up for dot coms”. The new version has an integrated in-memory text search engine, which is said to offer a ten-fold performance increase over existing techniques.

Version 7 has ‘deep’ XML integration facilities with intelligent searching and automated management with an integrated DB2 datatype. There is claimed easier access to heterogeneous data sources with an integrated distributed query capability and tools such as DB2 Relational Connect and an enhanced DB2 Data Links Manager.

The new database is said to be Windows 2000-ready with expanded OLE DB support and integration with Visual Studio development tools. It also comes with a Java Transaction API and it supports JDBC V2.

In terms of BI function, Version 7 has an integrated data warehouse centre with a claimed easy-to-use GUI and launchpad, heterogeneous source access, and industry-standard metadata management. There’s a new OLAP Starter Kit, based on Hyperion Essbase technology, and new query capabilities with relational OLAP functions in SQL.

For further information contact your local IBM representative.