July 2002

In this issue

3  XPERANTO – a marriage of XML and DB2
8  Introduction to DB2 UDB 7.2 Extender offerings
12 Generating DB2 utility jobs ASAP – part 2
27 Timeout/deadlock report
48 DB2 news

© Xephon plc 2002
Editor
Trevor Eddolls

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

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

© Xephon plc 2002. 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.
XPERANTO – a marriage of XML and DB2

INTRODUCTION

XPERANTO is a concatenation of X(ML) and (Es)peranto. Dr Zamenhof developed Esperanto as an international language in 1887! XML has become the standard data exchange language for Internet-based business applications, promising sizeable cost reductions by providing an automated and secure method for Internet data exchange. XPERANTO is IBM’s middleware technology for leveraging and accessing DB2.

FEATURES

XML features include:

1 Create XML views of relational data by automatically mapping the underlying relational database to a low-level XML view.

2 Create application-specific views using XQuery, which is a general-purpose declarative XML query language being standardized by W3C (World Wide Web Consortium).

3 Ability to query XML views of relational data subsets using XQuery. Only materializing the subset is an important benefit.

HIGH-LEVEL ARCHITECTURE

XPERANTO high-level architecture is illustrated in Figure 1.

XML VIEWS

XPERANTO creates default XML views from the target DB2 database. Users can define their own views by using XQuery, including views of views to reach higher levels of abstraction. Using XQuery is important because it is an XML standard, as opposed to proprietary languages like XSL-T, SilkRoute, Oracle XSQL, and Microsoft SQL Server. XQuery can process joins and recursions that some proprietary languages cannot.
IBM EXAMPLE

IBM uses the following purchase order relational database:

```
order
id  custname  custnum
10  Smith Construction  7734
9   Western Builders    7725

item
oid desc  cost
10 generator  8000
10 backhoe    24000

payment
oid due  amt
10 1/10/2001  20000
10 6/10/2001  12000
```

XPERANTO generates the following default XML view (oid is order id):

```
<db>
  <order>
    <row> <id> 10 </id> <custname> Smith Construction </custname> 
    <custnum> 7734 </custnum> </row>
  </order>
</item>
  <row> <oid> 10 </oid> <desc> generator </desc> <cost> 8000 </cost> 
</row>
```
<row> <oid> 10 </oid> <desc> backhoe </desc> <cost> 24000 </cost>
</row>

<payment>
  similar to <order> and <item>
</payment>
</db>

This is for the purchase order relational database with its default XML view.

Top-level elements are tables with their names as tags. Nested beneath are row elements with column names as tags and column values as text. XPERANTO captures primary and foreign key relationships in the XML schema (not shown).

Users would probably want to publish a list of orders. The default XML view is:

<order>
  <customer> Smith Construction </customer>
  <items>
    <item> <description> generator </description> <cost> 8000 </cost> <item>
    <item> <description> backhoe </description> <cost> 24000 </cost> <item>
      <payments>
        <payment due="1/10/2001"> <amount> 20000 </amount> </payment>
        <payment due="6/10/2001"> <amount> 12000 </amount> </payment>
      </payments>
    </item>
  </items>
</order>

<order>
  <customer> Western Builders </customer>
  ...
</order>

A user writes a user-defined XML view using XQuery:

1. create view orders as （
2.  for $order in view ("default")./order/row
3.  return
4.  <order>
5.  <customer> $order/custname </customer>
6.  <items>
7.    for $item in view("default")./item/row
8.      where $order/id = $item/oid
9.    return
10.  <item>
11.  <description> $item/desc </description> <cost> $item/cost </cost>
12.  <item>
13.  </items>  
14.  </payments>

15. for $payment in view("default"):item/row
16.     where $order/id = $payment/oid
17.     return
18. <payment due=$payment/date>
19.     <amount> $payment/amount </amount>
20. </payment> sortby@due)
21. </payments>
22. </order>
23. )

Lines 2-22 comprise an XQuery FLWR expression to construct each order element. In line 2, for causes variable $order to be bound to each order table row element and defines how to extract each row. The process starts at the default view, root, navigating to each order element nested under it, continuing to each row element nested under order elements. Line 8 predicate (order/id = $item/oid) creates a join of order with its items. Line 16 predicate ($order/id = $payment/oid) creates a join of order with its payments. Queries can be issued against the orders view:

1. for $order in view("orders")
2. let $items = $order/items
3. where $order/customer like "Smith%"
4. return $items

XPERANTO QUERY PROCESSING

Benefits of XPERANTO include producing only the desired relational data and pushing most memory and data-intensive computations down to DB2. XPERANTO query processing architecture is shown in Figure 2.

The processing steps for converting the query above include:

1. Parse XQuery into an intermediate representation known as XML Query Graph Model (XQGM).
2. XQGM combines the referenced XML views followed by rewrite optimizations to eliminate intermediate XML fragments.
3. Computation Pushdown splits the XQGM, capturing all memory and data-intensive processing and pushing it down to DB2; it also creates a tagger graph structure for use by Tagger Runtime to construct the XQuery result (tuples) in a single pass to return DB2 SQL to the user.
The DB2 SQL for XQuery is:

1. SELECT type, oid, desc, cost
2. FROM (SELECT Ø, order.id, order.custname, NULL, NULL
3. FROM order
4. WHERE order.custname LIKE "Smith%"
5. UNION ALL
6. SELECT 1, order.id, NULL, item.desc,
7. item.cost
8. FROM order, item
9. WHERE order.id = item.oid
10. ORDER BY oid, type, due

The SQL is a sorted outer union.

CONCLUSIONS

XML-based applications are imposing new requirements on DBMS, including publishing and querying existing relational data as XML. XPERANTO translates an XQuery into the correct DB2 SQL. XPERANTO performs optimization like view composition and pushing computation down to DB2.
Introduction to DB2 UDB 7.2 Extender offerings

With more and more projects involving Web-based transactions, the database you store your data on needs to be able to deal with new data formats (HTML, XML, etc) and requirements. DB2 uses a range of Extender offerings to meet these challenges. Full descriptions of each offering can be found in the respective Administration manuals, but all I want to do is give you an overview of each one.

There are five products in the DB2 Relational Extender range – Text Information Extender, Net.Search Extender, XML Extender, Image/Audio/Video Extender, and Spatial Extender.

TEXT INFORMATION EXTENDER
Text Information Extender (TIE) has the following main features:
Fuzzy searches – allow you to cater for searches where the user has made a ‘typo’ in the search string. For example, it will find ‘business’ when ‘businness’ is input.

Masking of individual or multiple characters in a search string. For example, it allows you to search for BUS% (which will find BUSiness and BUST), and BUS* (which will find words only four characters long beginning with BUS, ie BUST but not BUSINESS).

Allows you to search for two or more words, which then have to be in the same sentence/paragraph in the document.

Thesaurus searches – you can set up your own list of synonyms/relationships, based on your business needs. For example, you can set up a thesaurus that will treat DBA and D.B.A (and indeed DB.A!) as the same.

TIE queries can be incorporated into standard SQL, so it is possible to combine TIE predicates with other predicates.

As you can see, even this small subset of functions shows what a powerful offering TIE is—it will certainly allow you to get the most from your data.

There are design requirements which are needed to make the most of the offering, and these are all explained in the manual. The key piece of advice is, try to design your tables bearing in mind that you will be using TIE.

NET.SEARCH EXTENDER

The Net.Search Extender (NSE) offering was specifically developed for high-volume Web applications. It shares many features with TIE, but there are some important differences, in particular:

- NSE search is done through stored procedures, which means that you cannot add other predicates to the query.
- NSE does not have thesaurus support, ‘in same paragraph’ support, or ‘free text search’ support.

Net.Search has the ability to search for words, phrases, or fuzzy words. The index that is created for NSE entries is stored in memory, which makes access very fast.
The decision whether to use NSE or TIE depends on factors such as the volumes of data you will be processing, whether your table design favours one over the other, what type of search capability you need, etc.

**XML EXTENDER**

DB2 XML Extender allows you to store (decompose) and build (compose) XML type documents in DB2 tables. When you compose an XML document from data in DB2 tables, you can limit the documents you build based on standard SQL statements within the ‘build definition file’. I don’t want to get bogged down in the details here of how to do this, suffice it to say that it is possible and is a powerful feature. You can either store your XML documents in a single column (XMLColumn format) or break down the contents of the XML document and store the results in different columns of DB2 tables (XMLCollection format). The XMLColumn format is the easiest to implement, whereas the XMLCollection format provides greater flexibility.

Once you have decomposed your XML documents, you can use TIE to query them. This is a powerful combination of DB2 offerings.

**IMAGE/AUDIO/VISUAL EXTENDER**

Image/Audio/Visual Extender (IAV Extender) is like all the other Extender offerings, it uses User-Defined Types (UDTs), User-Defined Functions (UDFs), and triggers.

DB2 stores the input (image/video) in LOBs (large objects) in the database. These LOBs can be BLOBs (binary large objects), CLOBs (character large objects), or DBCLOBs (double-byte character large objects).

Because of the maximum possible size of each LOB (2GB), DB2 does not store the LOB in its tables, but stores a pointer to an area on disk external to DB2. This allows you to store up to 4TB of LOB space in a table (plenty for most people!). If the LOBs were stored in DB2, any inserts etc would have to be logged, and therefore you would require large (ie unmanageable) amounts of log space.

You obviously have to enable your database to be able to use the IAV extenders. This is detailed in the Administration manual, which is offloaded when you install the offering.
Some of the highlights of each component of the IAV offering are:

- **Image** – you can search for an image by content using a QBIC (Query by Image Content) catalog. This catalog stores information about the image (average colour, texture), so then you can search on that information.

- **Audio** – you can store and retrieve recordings in their entirety.

- **Video**:
  - you can search for a specific frame or for a scene change (where one frame changes drastically from the next frame) using a user-defined video index
  - you can store video clips and related information in DB2 tables, and then search the database using standard SQL.

**SPATIAL EXTENDER**

You can use the Spatial Extender offering to create a GIS (Geographic Information System). This GIS defines things like postcodes, road locations, buildings, etc. You can store this information in DB2 as information in rows. You can then query this information using standard SQL. An example of using spatial extender is when you phone a ‘store finder’ service, you give them your postcode, and they then find the shop closest to your location.

Continuing with the previous example, you need to somehow load the postcode information into DB2. There are tools/products available to do this. There are also tools available to display your output (the Spatial Extender Administration manual mentions ArcExplorer).

I hope I have given you a taster of what each of the DB2 Extender offerings can do for you. They are all very easy to install, and the test programs which come with them will give you a good demonstration of what the offerings have to offer.

Keep an eye on the IBM DB2 Web site for future offerings and enhancements to the current set.

C Leonard
Freelance Consultant (UK)

© Xephon 2002
Generating DB2 utility jobs ASAP – part 2

This month we conclude the code to copy or move data to different DB2 regions; mostly to testing subsystems but also sometimes to production.

Use the following JCL to run the REXX program in batch:

```
//myidLB JOB (GS,XWW,11000,4,12345),'SAM 12345',
   // CLASS=L,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=myid
/*
*ROUTE XEQ DB2C
*SPUFI EXEC PGM=IKJEFT1B,DYNAMBR=20,TIME=999
******************************************************************************
******************************************************************************
******************************************************************************
** EXAMPLES :
** EXEC DB2GEN(GENJCL) 'BU DB2C D11012 myid.TSNAME.DB2C.ODS +
   ** 70 YV21 T 0'
** EXEC DB2GEN(GENJCL) 'IC DB2C D11051 myid.TSNAME.NCP +
   ** 70 YV21'
** EXEC DB2GEN(GENJCL) 'IL DB2C D11051 myid.TSNAME.DM.D11051 +
   ** 70 YV2B'
** EXEC DB2GEN(GENJCL) 'BL DB2T D11131 myid.TSNAME.DMART +
   ** 70 YV2M''
******************************************************************************
******************************************************************************
******************************************************************************
//STEPLIB DD DISP=SHR,DSN=SAA.TS1.DB2APF
//SYSPROC DD DSN=SAA.DB2.CLISTLIB,DISP=SHR
//SYSTSPT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSDUMP DD SYSOUT=* 
//SYSTSin DD *
   EXEC DB2GEN(GENJCL) 'IU DB2C D12091 myid.TSNAME.DB2C.ODS +
      30 vs1 T X X '
/*
```

SAMPLE OUTPUT

```
IBMCOPY
```

```
//myidCP1 JOB (SG,XWW,11000,4,12345),'SAM 12345',
   // CLASS=F,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=myid
/*
*JOBPARM SYSASS=GSYS
*ROUTE XEQ DB2C
******************************************************************************
******************************************************************************
******************************************************************************
//CPY EXEC DB2UTIL,SYSTEM=DB2C,UID='''myidIC1'''
```
BMC UNLOAD

myidBU1 JOB (SG,XWW,11000,4,12345), 'SAM 12345',
   CLASS=F,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=myid
   */
   **JOPARM SYSAFF=SYS
   */ROUTE XEO DB2C
   ********************************************
   **BMCL proc TNAME=',UNIT='CART',LBL=',VOLREF=
   **UL EXEC PGM=ADUMAIN,COND=(8,LT,UNCAT),REGION=OM,
   ** PARM='DB2C,myidBU1,NEW/RESTART,,MSGLEVEL(1)'
   */
   **STEPLIB DD DSN=SAA.TS1.DB2APF,DISP=SHR
   **SYSPRINT DD SYSOUT=* 
   **SYSOUT DD SYSOUT=* 
   **UPRINT DD SYSOUT=* 
   **SYSDUMP DD SYSOUT=* 
   **SYSREC DD DSN=TEST.VV1.DB2C.&TNAME..BMCL.D01091,
IBM UNLOAD

//myidIU1 JOB (SG,XWW,11000,4,12345),'SAM 12345',
// CLASS=F,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=myid
/**
//*JOBPARM SYSAFF=GSYS
/*/ROUTE XEO DB2C

******************************************************************************
//UL1 EXEC PGM=IKJEFT01,REGION=2048K
//STEPLIB DD DSN=SAA.TS1.DB2APF,DISP=SHR
//SYSTSPRT DD SYSOUT=*
IBM LOAD

//myidLD1 JOB (SG,XWW,11000,4,12345),'SAM 12345',
//       CLASS=L,MSGCLASS=T,       
//       MSGLEVEL=(1,1),NOTIFY=myid
//*
/*ROUTE XEQ DB2T
//IBMLD  PROC TNAME=
//LD EXEC DB2UTIL,SYSTEM=DB2T,UID=''myidLD1''
/**
//SYSREC DD DSN=TEST.VV1.DB2C.&TNAME..BMGUL.D01091,DISP=SHR
//SYSUTI DD DSN=TEST.VV1.DB2C.&TNAME..SYSUTI,
//     DISP=(NEW,DELETE,DELETE),UNIT=SYSDA,
//     SPACE=(CYL,(5,200),RLSE)
//SORTOUT DD DSN=TEST.VV1.DB2C.&TNAME..SORTOUT,
//     DISP=(NEW,DELETE,DELETE),UNIT=SYSDA,
//     SPACE=(CYL,(5,200),RLSE)
//SYSIN DD DSN=TEST.VV1.DB2C.LOADCNTL.D01091(&TNAME.),
//     DISP=SHR
//PEND
/**

***********************************************************************
//ST1 EXEC IBMLD,TNAME='ACCT'
//ST2 EXEC IBMLD,TNAME='ACCT_KIND'
//ST3 EXEC IBMLD,TNAME='ACTIVITY'

***********************************************************************
**BMC LOAD**

```bash
//myidBL1 JOB (SG,XWW,11000,4,12345),'SAM 12345',
// CLASS=L,MSGCLASS=T,
// MSGLEVEL=(1,1),NOTIFY=myid
/**
/*ROUTE XEQ DB2T
//BMCLD PROC TNAME=
//LOAD1 EXEC PGM=AMUUMAIN,
// PARM='DB2T,myidBL1,NEW/RESTART,,MSGLEVEL(1)'
//STEP1B DD DSN=SAA.TSI.DB2APF,DISP=SHR
/**
//SYSREC00 DD DSN=TEST.VV1.DB2C.&TNAME..BMCUL.D01091,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//UTPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSERR DD DISP=(NEW,CATLG,CATLG),SPACE=(CYL,(1,1),RLSE),
// DSN=TEST.BMCUL.&TNAME..SYSERR.D01091,UNIT=SYSDA
//SYSDISC DD DISP=(NEW,CATLG,CATLG),SPACE=(CYL,(2,3),RLSE),
// DSN=TEST.BMCUL.&TNAME..SYSDISC.D01091,UNIT=SYSDA
//SORTWK01 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK01 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK02 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK03 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK04 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK05 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK06 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK07 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK08 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK09 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK10 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK11 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SORTWK12 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,50))
//SYSIN DD DSN=TEST.VV1.DB2C.LOADCNTL.D01091(&TNAME.),DISP=SHR
//PEND
/**
*****************************************************************************
//ST1 EXEC BMCLD,TNAME='ACCT'
//ST2 EXEC BMCLD,TNAME='ACCTKIND'
//ST3 EXEC BMCLD,TNAME='ACTIVITY'
*****************************************************************************
```

**COMPARE DDL FOR TABLES AND COLUMNS**

In my current data warehouse project we have seven DB2 regions, each with its own special purpose. One of the DBA’s responsibilities is to implement database changes such as adding a new column, removing columns, and changing column attributes.
It is important for the changes to be made correctly in all of the DB2 regions. A DBA cannot be too careful about keeping the tables and columns in sync between different regions, as problems can occur at any time during an application’s run time or DBA’s utility job if there are unmatching tables or columns.

The following REXX program can be used to check the synchronization of the tables and columns between two different DB2 subsystems:

```rexx
/******************** REXX **************************/
/** Compare the DDL for tables and columns between two subsystems. */
/** SQLX, an in-house developed REXX/SQIL interface is used, but */
/** the program will still work without SQLX, by providing the */
/** input files that are output from the SQL: */
/** SELECT TBNAME, COLNO, NAME, COLTYPE, LENGTH, SCALE, NULLS, */
/** DEFAULT, DEFAULTVALUE */
/** FROM SYSIBM.SYSCOLUMNS */
/** WHERE TBCREATOR = 'creator' */
/** ORDER BY TBNAME, COLNO */
/******************** **************************/
/** Input parms */
/** subsys1 - DB2 subsystems 1 */
/** subsys2 - DB2 subsystems 2 */
/** creator1 - Creator of the tables in subsystem 1 */
/** creator2 - Creator of the tables in subsystem 2 */
/** indsn1 - Input dataset for the objects from subsys1 */
/** indsn2 - Input dataset for the objects from subsys2 */
/******************** **************************/
Arg subsys1 indsn1 creator1 subsys2 indsn2 creator2
x = sysdsn(insn1)
If x <> 'OK'
  Then call newalloc indsn1 indd1
Else do
  Address tso
  "ALLOC DDNAME(indd1) DSNAME('"insn1"') SHR REUSE ",
  " RECFM(F B) LRECL(132) BLKSIZE(13200)"
End
y = sysdsn(insn2)
If y <> 'OK'
  Then call newalloc indsn2 indd2
Else do
  Address tso
  "ALLOC DDNAME(indd2) DSNAME('"insn2"') SHR REUSE ",
  " RECFM(F B) LRECL(132) BLKSIZE(13200)"
End

"SQLX CONNECT "subsys1
If SQL_CC <> Ø then do
```

rc = SQL_ERROR()
say "*** Connection error to" subsys1
say "*** INDSN1 file will be used as it is ...."
End
Else do
stmt1 = "SELECT",
    " TBNAME, COLNO, NAME, COLTYPE, LENGTH, SCALE, NULLS",
    " DEFAULT, DEFAULTVALUE",
    " FROM SYSIBM.SYSCOLUMNS",
    " WHERE TBCREATOR = "creator1"
    " ORDER BY TBNAME, COLNO"
"SQLX DECLARE D1 CURSOR FOR "stmt1
If SQL_CC <> Ø then do
rc = SQL_ERROR()
say "*** SQL execution error in" subsys1
say "*** INDSN1 file will be used as it is...."
End
Else do
recl. = '
Do i = 1 to D1
  recl.i = D1_TBNAME.i D1_COLNO.i D1_NAME.i D1_COLTYPE.i,
    D1_LENGTH.i D1_SCALE.i,
    D1_NULLS.i D1_DEFAULT.i D1_DEFAULTVALUE.i
End
Address TSO
"EXECIO * DISKW indd1 (stem recl. finis"
End
End

"SQLX CONNECT "subsys2
If SQL_CC <> Ø then do
rc = SQL_ERROR()
say "*** Connection error to" subsys2
say "*** INDSN2 file will be used as it is...."
End
Else do
stmt2 = "SELECT",
    " TBNAME, COLNO, NAME, COLTYPE, LENGTH, SCALE, NULLS",
    " DEFAULT, DEFAULTVALUE",
    " FROM SYSIBM.SYSCOLUMNS",
    " WHERE TBCREATOR = "creator2"
    " ORDER BY TBNAME, COLNO"
"SQLX DECLARE D2 CURSOR FOR "stmt2
If SQL_CC <> Ø then do
rc = SQL_ERROR()
say "*** SQL execution error in" subsys2
say "*** INDSN2 file will be used as it is ...."
End
Else do
rec2. = '

Do i = 1 to D2
    rec2.i = D2_TBNNAME.i D2_COLNO.i D2_NAME.i D2_COLTYPE.i,
             D2_LENGTH.i D2_SCALE.i,
             D2_NULLS.i D2_DEFAULT.i D2_DEFAULTVALUE.i
End
Address TSO
"EXECIO * DISKW indd2 (stem rec2. finis"
End
End

Number_of_lines1 = Ø
Number_of_lines2 = Ø
Number_of_tables1 = Ø
Number_of_tables2 = Ø
Number_of_cols1. = Ø
Number_of_cols2. = Ø
tnames1. = ''
tnames2. = ''
cnames1. = ''
cnames2. = ''
cnames1. = ''
cnames2. = ''
clength1. = ''
clength2. = ''
cscales1. = ''
cscales2. = ''
cnulls1. = ''
cnulls2. = ''
cdflts1. = ''
cdflts2. = ''
cdvalue1. = ''
cdvalue2. = ''
diffstmt_table. = ''
diffstmt_numcol. = ''
diffstmt_cname. = ''
diffstmt_dtyp. = ''
diffstmt_dlen. = ''
diffstmt_scale. = ''
diffstmt_null. = ''
diffstmt_def. = ''
diffstmt_defv. = ''
dt = Ø
dnc = Ø
dcn = Ø
dct = Ø
d1 = Ø
ds = Ø
dn = Ø
dd = Ø
dv = Ø
/

/**************************************************************************/
/* Read in the input - sysibm.syscolumns                        */
/**************************************************************************/
"EXECIO * DISKR indd1 (FINIS"
Number_of_lines1 = queued()
number_of_columns = 1
i = Ø
Do Number_of_lines1
  Pull w1 w2 w3 w4 w5 w6 w7 w8 w9
  If w2 = 1 then do
    i = i + 1
    tnames1.i = w1           /* i-th table in subsys 1 */
    cnames1.i.1 = w3         /* 1st column of i-th table in sybsys 1 */
    ctypes1.i.1 = w4
    clength1.i.1 = w5
    cscales1.i.1 = w6
    cnulls1.i.1 = w7
    cdfilts1.i.1 = w8
    cdvalue1.i.1 = w9
  end
  else if w2 <> ' ' then do
    Number_of_cols1.i = w2
    cnames1.i.w2 = w3       /* w2-th col. of i-th table in sybsys 1 */
    ctypes1.i.w2 = w4
    clength1.i.w2 = w5
    cscales1.i.w2 = w6
    cnulls1.i.w2 = w7
    cdfilts1.i.w2 = w8
    cdvalue1.i.w2 = w9
  end
else nop
End
Number_of_tables1 = i
say subsys1 'has' Number_of_tables1 'tables.'

"EXECIO * DISKR indd2 (FINIS"
Number_of_lines2 = queued()
j = Ø
Do Number_of_lines2
  Pull w1 w2 w3 w4 w5 w6 w7 w8 w9
  If w2 = 1 then do
    j = j + 1
    tnames2.j = w1
    cnames2.j.1 = w3
    ctypes2.j.1 = w4
    clength2.j.1 = w5
    cscales2.j.1 = w6
    cnulls2.j.1 = w7
    cdfilts2.j.1 = w8

cdvalue2.j.1 = w9
end
else if w2 <> '' then do
  Number_of_cols2.j = w2
cnames2.j.w2 = w3
ctypes2.j.w2 = w4
clength2.j.w2 = w5
cscales2.j.w2 = w6
cnulls2.j.w2 = w7
cdfilts2.j.w2 = w8
cdvalue2.j.w2 = w9
end
else nop
end
Number_of_tables2 = j
say subsys2 'has' Number_of_tables2 'tables.'
say ''
say left(subsys1,25) subsys2
say left('-------',25) '-------'
j = 1
j = 1
Do until ((i > Number_of_tables1) | (j > Number_of_tables2))
  if tnames1.i < tnames2.j then do
    say left(tnames1.i,25) '***********'
i = i + 1
  end
  else if tnames1.i > tnames2.j then do
    say left('***********',25) tnames2.j
    j = j + 1
  end
  else do
    say left(tnames1.i,25) tnames2.j
    i = i + 1
    j = j + 1
  end
end
If i > Number_of_tables1 then
  Do while (j <= Number_of_tables2)
    say left('***********',25) tnames2.j
    j = j + 1
  end
else if j > Number_of_tables2 then
  Do while (i <= Number_of_tables1)
    say left(tnames1.i,25) '***********'
i = i + 1
  end
  else nop
say ''
say subsys1 i-1 'tables ' subsys2 j-1 'tables '
/* Compare the columns of the tables */
compare_table:
   i = 1
   j = 1
Do while (( i < Number_of_tables1) & (j < Number_of_tables2))
   If tnames1.i < tnames2.j then do
      dt = dt + 1
      diffstmt_table.dt = tnames1.i ' is missing in ' subsys2
      i = i + 1
   end
   else if tnames1.i > tnames2.j then do
      dt = dt + 1
      diffstmt_table.dt = tnames2.j ' is missing in ' subsys1
      j = j + 1
   end
   else do /* same table names */
      call compare_column
      i = i + 1
      j = j + 1
   end
End

If i = Number_of_tables1 then
   if j = Number_of_tables2 then nop
else do until (j > Number_of_tables2)
   j = j + 1
   dt = dt + 1
   diffstmt_table.dt = tnames2.j ' is missing in ' subsys1
end
else if j = Number_of_tables2 then nop
else do until (i > Number_of_tables1)
   i = i + 1
   dt = dt + 1
   diffstmt_table.dt = tnames1.i ' is missing in ' subsys2
end

/** Write the report */
Say ' '
If dt > 0 then
   Do i = 1 to dt
      Say diffstmt_table.i
   End
Else nop
Say ' '
If dnc > 0 then
   Do i = 1 to dnc

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
Say diffstmt_numcol.i
End
Else nop
Say ' '
If dcn > 0 then
  Do i = 1 to dcn
    Say diffstmt_cname.i
  End
Else nop
Say ' '
If dct > 0 then
  Do i = 1 to dct
    Say diffstmt_dtyp.i
  End
Else nop
Say ' '
If d1 > 0 then
  Do i = 1 to d1
    Say diffstmt_dlen.i
  End
Else nop
Say ' '
If ds > 0 then
  Do i = 1 to ds
    Say diffstmt_scale.i
  End
Else nop
Say ' '
If dn > 0 then
  Do i = 1 to dn
    Say diffstmt_null.i
  End
Else nop
Say ' '
If dd > 0 then
  Do i = 1 to dd
    Say diffstmt_def.i
  End
Else nop
Say ' '
If dv > 0 then
  Do i = 1 to dv
    Say diffstmt_defv.i
  End
Else nop

Address tso
"FREE FI(indd1)"
"FREE FI(indd2)"
exit (Ø)
compare_column:
/*********************/
/* Compare the columns of the same tables from the two subsystems */
/*  cnames1.i.k : k-th column of the i-th table in subsys 1 */
/*  cnames2.j.k : k-th column of the j-th table in subsys 2 */
/*********************/
If Number_of_cols1.i <> Number_of_cols2.j then do
  dnc = dnc + 1
  diffstmt_numcol.dnc =,
    tnames1.i 'has ' Number_of_cols1.i 'cols in' subsys1||',',
    Number_of_cols2.j 'cols in' subsys2
  End
else nop
  k = 1
  l = 1
Do until ((k > Number_of_cols1.i) | (l > Number_of_cols2.j))
  If cnames1.i.k = cnames2.j.l then do
    call compare_datatype
    call compare_datatype
    call compare_length
    call compare_scale
    call compare_null
    call compare_default
    call compare_defvalue
    k = k + 1
    l = l + 1
    End
  Else do
    dcn = dcn + 1
    diffstmt cname.dcn =,
      tnames1.i k cnames1.i.k ' in ' subsys1 not matching
    leave
    End
  End
  return Ø

compare_datatype :
If ctypes1.i.k <> ctypes2.j.l then do
  dcp = dct + 1
  diffstmt_dtyp.dct =,
    tnames1.i 'in' subsys1 'has unmatching data types for column',
    k cnames1.i.k
  End
else nop
  return

compare_length :
If clength1.i.k <> clength2.j.l then do
  dl = d1 + 1
  diffstmt_clen.dl =,
    tnames1.i 'in' subsys1 'has unmatching column length for',

k cnames1.i.k
End
else nop
return

compare_scale :
If cscales1.i.k <> cscales2.j.l then do
  ds = ds + 1
  diffstmt_scale.ds =,
      tnames1.i 'in' subsys1 'has unmatching column scale for',
      k cnames1.i.k
End
else nop
return

compare_null :
If cnulls1.i.k <> cnulls2.j.l then do
  dn = dn + 1
  diffstmt_null.dn =,
      tnames1.i 'in' subsys1 'has unmatching nullable value for',
      k cnames1.i.k
End
else nop
return

compare_default :
If cdflts1.i.k <> cdflts2.j.l then do
  dd = dd + 1
  diffstmt_def.dd =,
      tnames1.i 'in' subsys1 'has unmatching default indicator for',
      k cnames1.i.k
End
else nop
return

compare_defvalue :
If cdvalue1.i.k <> cdvalue2.j.l then do
  dv = dv + 1
  diffstmt_defvdv =,
      tnames1.i 'in' subsys1 'has unmatching default value for',
      k cnames1.i.k
End
else nop
return

newalloc: procedure
  arg dsn ddn
  Address TSO
    "ALOC DDNAME("ddn") DSNAMES('"dsn"') NEW CATALOG SPACE(5,5) TRACKS",
    " UNIT(DISK) RECFM(F B) LRECL(132) BLKSIZE(13200)"
return
SQL_ERROR:
   SAY "SQL_DB2SSN = ''SQL_DB2SSN''"
   SAY "SQL_CC = ''SQL_CC''"
   SAY "SQL_RC = ''SQL_RC''"
   SAY "SQL_REASON = ''SQL_REASON''"
   SAY "SQL_MESSAGE = ''SQL_MESSAGE''"
RETURN 0

Use the following JCL to run the REXX program in batch. The REXX program is located in myid.DB2GEN.CLIST.

```plaintext
//myidLB JOB (GS,XWW,11000,4,12345),'SAM 12345',
   // CLASS=L,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=myid
   //*
   /*ROUTE XEQ DB2C
   //SPUFI EXEC PGM=IKJEFTIB,DYNAMNBR=20,TIME=999
   //STEPlib DD DISP=SHR,DSN=SAA.TS1.DB2APF
   //SYSPROC DD DSN=SAA.DB2.CLISTLIB,DISP=SHR
   //SYSTSPRT DD SYSOUT=* 
   //SYSPRINT DD SYSOUT=* 
   //SYSUDUMP DD SYSOUT=* 
   //SYSTSIN DD *
      EXEC DB2GEN(CMPDDL) 'DB2A myid.SYSCOL.DB2A VV1 + DB2B myid.SYSCOL.DB2B VV1'
   /*
```

The output from this program will look like the following:

```
DB2A has 477 tables.
DB2B has 474 tables.

<table>
<thead>
<tr>
<th>DB2A</th>
<th>DB2B</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT</td>
<td>ACCT</td>
</tr>
<tr>
<td>ACCT_STRUCT</td>
<td>ACCT_STRUCT</td>
</tr>
<tr>
<td>ACTY_TRCKNG</td>
<td>ACTY_TRCKNG</td>
</tr>
<tr>
<td>ADDR</td>
<td>ADDR</td>
</tr>
<tr>
<td>ADDR_REL</td>
<td>ADDR_REL</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>MSC_CAP_MAP</td>
<td></td>
</tr>
<tr>
<td>MSC_MED_MAP</td>
<td></td>
</tr>
<tr>
<td>MUST_OFFR_BEN_PLN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>WH_SRC</td>
<td>WH_SRC</td>
</tr>
<tr>
<td>ZIP_CD</td>
<td>ZIP_CD</td>
</tr>
</tbody>
</table>

DB2A 477 tables  DB2B 474 tables

MSC_CAP_MAP is missing in DB2A
MSC_MED_MAP is missing in DB2A
```
Many DB2 administrators and DB2 application programmers have wondered how they can find out which transactions have timed out and which ones haven’t, how often these transactions have timed out, which plan was timed out, and which resources (database, tablespace) were affected. All this information and more is stored in the DB2 master address space log, but it is not formatted and is too long. The following REXX program analyses the DB2 master log and looks for timeout/deadlock messages and related messages.

This REXX discovers and converts the many timeout messages in the log to one summary record. For example DSNT376I indicates a timeout message and a subsequent DSNT500I message indicates resource information about the timed-out transactions. Message codes DSNT375I, DSNT376I, and DSNT500I may not appear in order in the DB2 log. Therefore corresponding messages are found and merged in this REXX program. Message code DSNT500I can include resource names or one of dbid, obid, psid, and isobid. For rows containing dbid, psid, obid, and isobid, a program has been written (DBAB101) to convert dbid, psid, obid, and isobid to database name and tablespace name.

You can see sample timeout/deadlock reports at the end of the article. Here are descriptions of the programs and JCL:
• DB2LOGA – analyse DB2 logs and write each corresponding timeout record to the output file.
• DBAB101 – convert (replace) dbid, psid, obid, isobid to database name and tablespace name in TIMEOUT_REPORT1 table.
• DDL – DDLs of PDBA.TIMEOUT_REPORT1 table.
• DB2LOGAJ – JCL to run these programs and print sample timeout/deadlock report using the DSNTEP2 program.
• CPLIDB2 – sample compile job for DB2-vapli-batch programs.

DB2LOGA REXX PROGRAM

/* rexx */
/*****************************/
/* main */
/* This program analyses DB2 master log to find which */
/* transactions or jobs timed out */
/* Also including some information about time out tran. */
/* and holder. These are following. */
/* .Timed out transaction correlation-id. */
/* .Timed out transaction connection-id */
/* .Timed out transaction date and time. */
/* .holder transaction correlation-id. */
/* .holder transaction connection-id */
/* .Which resource (database and tablespace name) */
/* .object type */
/* .reason code */
/* .message code (deadlock or timeout) */
/*****************************/
arg dbmid
ALLOC F(D2LOG) DA(syspdba.ps0.$dbmid$.log.temp) SHR$
eof = '0'
wmmsg_flag = '0'
row = 0
i1 = 0
i2 = 0
EXECIO 1 DISKR DB2LOG$
do while eof = '0'
pull line
row = row + 1
if wmsg_flag = '0'
   then wmsg = word(line,3)
if word(line,3) = '——' & word(line,7) > '1999' then do
   wdate = substr(line,36,11)
say wdate
   /* add output */
do
end
if wmsg = 'DSNT375I' & wmsg_flag = '0' then do
  wmsg_flag = '1'
  wmsg_row_lmt = 13
  row_x = ∅
  i1 = i1 + 1
  call init_array
end
if wmsg = 'DSNT376I' & wmsg_flag = '0' then do
  wmsg_flag = '1'
  wmsg_row_lmt = 13
  row_x = ∅
  i1 = i1 + 1
  call init_array
end
if wmsg = 'DSNT501I' & wmsg_flag = '0' then do
  wmsg_flag = '1'
  row_x = ∅
  wmsg_row_lmt = 7
  w_find = '0'
end
select
  when wmsg = 'DSNT375I' then
    call dsnt375i
  when wmsg = 'DSNT376I' then
    call dsnt376i
  when wmsg = 'DSNT501I' then
    call dsnt501i
  otherwise nop
end
$execio 1 diskr DB2Log$
if word(line,2) = '//STARTING' then
  eof = '1'
if rc > ∅ then eof = '1'
end
$execio 0 diskr DB2log (finis$
$free file(DB2log)$
$delete syspdba.ps∅.dbmid$.timeout$
$alloc da(syspdba.ps∅.dbmid$.timeout) new cylinders space(5,5)
  lrecl(132) blksize(13200) dsorg(ps) recfm(f,b) $'
$alloc fi(DB2logo) da(syspdba.ps∅.dbmid$.timeout) shr$
push ''
mon.1 = 'JAN'
mon.2 = 'FEB'
mon.3 = 'MAR'
mon.4 = 'APR'
mon.5 = 'MAY'
mon.6 = 'JUN'
mon.7 = 'JUL'
mon.8 = 'AUG'
mon.9 = 'SEP'
mon.10 = 'OCT'
mon.11 = 'NOV'
mon.12 = 'DEC'
do i = 1 to 10
  if w_mesaj.i = 'DSNT376I' then
    ww_mesaj = 'TIMEOUT'
  if w_mesaj.i = 'DSNT375I' then
    ww_mesaj = 'DEADLOCK'
  do m = 1 to 12
     if word(w_date1.i,2) = mon.m
       then leave
  end
  if m < 10 then
    ww_date = word(w_date1.i,3)||’-0’||m||’-’||word(w_date1.i,1)
  else
    ww_date = word(w_date1.i,3)||’-’||m||’-’||word(w_date1.i,1)
    ww_time = translate(w_time1.i,’:\’||’.’
    line1 = w_plan1.i ||’ ’|| w_cor1.i ||’ ’|| w_conn1.i ||’ ’
    line1 = line1 || ww_date ||’ ’|| ww_time ||’ ‘
    line1 = line1 || w_plan2.i ||’ ’|| w_cor2.i ||’ ‘
    line1 = line1 || w_conn2.i ||’ ’|| w_dbid2.i ||’ ‘
    line1 = line1 || w_object1.i ||’ ’|| w_mesaj ||’ ‘
    line1 = line1 || w_reason1.i ||’ ’|| w_type1.i
  end
  say length(line1)
  if length(line1) > 125 then do
    push line1
    $execio 1 diskw DB2logo $
  end
$execio 0 diskr DB2logo (finis$ $free file(DB2logo)$
return
/***************************************************************************/
/* dsnt376i */
/***************************************************************************/

dsnt375i:
  row_x = row_x + 1
  if row_x = 1 then do
    w_mesaj.11 = 'DSNT375I'
    w_plan1.11 = substr(word(line,5),6,8)
    w_date1.11 = wdate
    w_time1.11 = word(line,1)
  end
  if row_x < 5 then do
    if substr(line,29,4) = 'CORR' then
      w_cor1.11 = substr(line,44,12)
    if substr(line,29,4) = 'CONN' then
      w_conn1.11 = substr(line,43,8)
end
if substr(line,29,8) = 'IS DEADL' then
    w_plan2.il = substr(word(line,4),6,8)
if row_x > 5 then
do
    if substr(line,29,4) = 'CORR' then
        w_corr2.il = substr(line,44,12)
    if substr(line,29,4) = 'CONN' then
        w_conn2.il = substr(line,43,8)
end
if word(line,2) = 'MEMBER' then
    w_dbid2.il = word(line,3)
    wmsg_flag = 'Ø'
    wmsg = ''
end
return
*******************************************************************************/
/* dsnt376i  */
*******************************************************************************/
dsnt376i:
row_x = row_x + 1
if row_x = 1 then
do
    w_mesaj.il = 'DSNT376I'
    w_plan1.il = substr(word(line,5),6,8)
    w_date1.il = wdate
    w_timel.il = word(line,1)
end
if row_x < 5 then
do
    if substr(line,29,4) = 'CORR' then
        w_corr1.il = substr(line,44,12)
    if substr(line,29,4) = 'CONN' then
        w_conn1.il = substr(line,43,8)
end
if substr(line,29,8) = 'IS TIMED' then
    w_plan2.il = substr(word(line,10),6,8)
if row_x > 5 then
do
    if substr(line,29,4) = 'CORR' then
        w_corr2.il = substr(line,44,12)
    if substr(line,29,4) = 'CONN' then
        w_conn2.il = substr(line,43,8)
end
if word(line,2) = 'MEMBER' then
    w_dbid2.il = word(line,3)
    wmsg_flag = 'Ø'
    wmsg = ''
end
return
/*******************************/
/* dsnt501i */
/*******************************/
dsnt501i:
  row_x = row_x + 1
  if substr(line,32,4) = 'CORR' then
    w_corr = substr(line,47,12)
    if substr(line,32,4) = 'CONN' then
      do
        w_conn = substr(line,46,8)
        do ii = i1 to 1 by -1
          if w_corr1.ii = w_corr &,
            w_conn1.ii = w_conn then
            w_find = '1'
          if w_find = '1' then leave
        end
      end
  /* say w_corr  w_conn  w_find */
  pull a
  if a = '' then exit */
  if w_find = '1' then
    do
      if word(line,1) = 'REASON' then
        w_reason = word(line,2)
      if word(line,1) = 'TYPE' then
        w_type = word(line,2)
      if word(line,1) = 'NAME' then
        do
          w_object1.ii = substr(line,37,17)
          wmsg_flag = '0'
          wmsg = ''
        end
      if row_x = 7 then do
        wmsg_flag = '0'
        wmsg = ''
      end
    end
  return
/*******************************/
/* init_array */
/*******************************/
init_array:
  w_plan1.ii = '
  w_date1.ii = '
  w_time1.ii = '
  w_corr1.ii = '
  w_conn1.ii = '
  w_plan2.ii = '
  w_corr2.ii = '
  w_conn2.ii = ''
w_dbid2.i1 = ' ',
  w_reason1.i1 = ' ',
  w_typel.i1 = ' ',
  w_object1.i1 = ' ',

return
/**
 ******************************************
 /* syspdba.ps0.dbp2.timeout */
/* */
/* */
line1 = w_planl.i || ' ' || w_corry.i || ' ' || w_conn1.i || ' ' || w_reason1.i || ' ' || w_correct2.i || ' ' || w_dbid2.i || ' ' || w_object1.i || ' ' || w_message || ' ' || w_typel.i
/* */ 8 Plan1 (timed out plan)
/* */ 12 Corr.id.1
/* */ 8 conn.id.1
/* */ 10 date
/* */ 8 time
/* */ 8 Plan2 (holder plan)
/* */ 12 corr.id.2
/* */ 8 conn.id.2
/* */ 4 dbid member
/* */ 17 object name
/* */ 8 message code
/* */ 8 reason code
/* */ 8 object type
******************************************/

DBAB101 VA-PLI PROGRAM

DBAB101: PROC OPTIONS(MAIN);
/* ---------------------------------------------- */
/* PROGRAM NAME : DBAB101 */
/* JCL NAME : DBAB101 */
/* DATE : 09.09.2001 */
/* DESCRIPTION : UPDATE DBAB101_REPORT1 TABLE */
/* */
/* DESCRIPTION : CHECKS THE DATABASE NAME AND TABLESPACE */
/* */
/* NAME IN TIMEOUTREPORT1 TABLE. IF THE */
/* NAMES INCLUDE ANY OF DBID,PSID, OR OBD */
/* THESE COLUMNS ARE UPDATED FROM CATALOG */
/* TABLES WITH CORRESPONDING DBNAME AND */
/* TSNAME. */
/* */
DCL ( ADDR,ONCODE,NULL,DATE,MOD,DATETIME,ABS,
     SUBSTR,TRANSLATE,VERIFY ) BUILTIN ;
DCL 1 SYSDB,
  5 DB_NAME CHAR(8),
  5 DB_CREATOR CHAR(8),
5 DB_DBID  BIN FIXED(15);

DCL 1 SYSTS,
  5 TS_NAME  CHAR(8),
  5 TS_CREATOR  CHAR(8),
  5 TS_DBNAME  CHAR(8),
  5 TS_DBID  BIN FIXED(15),
  5 TS_OBJID  BIN FIXED(15),
  5 TS_PSID  BIN FIXED(15);

DCL 1 SYSTB,
  5 TB_NAME  CHAR(18) VAR,
  5 TB_CREATOR  CHAR(8),
  5 TB_DBNAME  CHAR(8),
  5 TB_TSNAME  CHAR(8),
  5 TB_DBID  BIN FIXED(15),
  5 TB_OBJID  BIN FIXED(15);

DCL 1 SYSIX,
  5 IX_NAME  CHAR(18) VAR,
  5 IX_CREATOR  CHAR(8),
  5 IX_TBLNAME  CHAR(18) VAR,
  5 IX_TBCREATOR  CHAR(8),
  5 IX_DBID  BIN FIXED(15),
  5 IX_OBJID  BIN FIXED(15),
  5 IX_ISOBSID  BIN FIXED(15),
  5 IX_DBNAME  CHAR(8),
  5 IX_INDEXSPACE  CHAR(8);

DCL 1 DCLTIMEOUT_REPORT11,
  5 TM_PLAN  CHAR(8),
  5 TM_PLAN_CORR_ID  CHAR(12),
  5 TM_PLAN_CONN_ID  CHAR(8),
  5 TM_DATE  CHAR(10),
  5 TM_TIME  CHAR(8),
  5 TM_PLAN_HOLDER  CHAR(8),
  5 TM_PLANHD_CORR_ID  CHAR(12),
  5 TM_PLANHD_CONN_ID  CHAR(8),
  5 TM_DB2_MEMBER  CHAR(4),
  5 TM_DB2_DATABASE  CHAR(8),
  5 TM_DB2_TABLESPACE  CHAR(8),
  5 TM_MESSAGE_CODE  CHAR(8),
  5 TM_DB2_REASON_CODE  CHAR(8),
  5 TM_DB2_OBJ_TYPE  CHAR(8);

DCL WS_PIC1  PIC'(8)9';
DCL WS_DEC1  DEC FIXED(8,0);
DCL WS_BINDB  BIN FIXED(15);
DCL WS_BINTS  BIN FIXED(15);
DCL WS_RCODE  CHAR(01) INIT('0');
DCL WS_DB_NAME  CHAR(08) INIT(' '); 
DCL WS_TS_NAME  CHAR(08) INIT(' ');
DCL WS_SQLCODE  PIC'S9999' INIT('0');
DCL WS_C_COUNT  PIC'(5)9' INIT(0);
DCL WS_C1  PIC'(8)9' INIT(0);
DCL WS_C2  PIC'(8)9' INIT(0);
EXEC SQL INCLUDE SQLCA ;
/****************************************************************************/
/*  MAIN PROCEDURE */
/*
*  SELECT ROWS FROM TIMEOUT_REPORT1 TABLE
*  IF DB2_TABLESPACE OR DB2_DATABASE COLUMNS
*  GREATER THAN 0, THESE COLUMNS WILL
*  BE UPDATED. THESE INCLUDE DBID, PSID, OBID
*  ISOBID
***************************************************************************/
ON ERROR
BEGIN;
  PUT DATA ;
  EXEC SQL
  ROLLBACK;
  STOP
END ;
EXEC SQL
DECLARE C01 CURSOR WITH HOLD FOR
SELECT TM_DB2_DATABASE ,
TM_DB2_TABLESPACE ,
TM_DB2_REASON_CODE ,
TM_DB2_OBJ_TYPE
FROM PDBA.TIMEOUT_REPORT1
WHERE TM_DB2_DATABASE > '0'
   OR TM_DB2_TABLESPACE > '0'
FOR UPDATE OF TM_DB2_DATABASE,
    TM_DB2_TABLESPACE ;
EXEC SQL
OPEN C01 ;
IF SQLCODE < 0 THEN
  CALL SQL_ERROR;
EXEC SQL
FETCH C01 INTO :TM_DB2_DATABASE,:TM_DB2_TABLESPACE,
   :TM_DB2_REASON_CODE,:TM_DB2_OBJ_TYPE;
IF SQLCODE < 0 THEN
  CALL SQL_ERROR;
DO WHILE(SQLCODE = 0) ;
  WS_C2 = WS_C2 + 1;
  CALL CHECK_DB_TS;
  PUT SKIP EDIT
      (TM_DB2_DATABASE,WS_DB_NAME,TM_DB2_TABLESPACE,WS_TS_NAME)
      (A,X(2),A,X(2),A,X(2),A);
  IF WS_RCODE = '1' THEN
    CALL UPDATE_TABLE;
  EXEC SQL
    FETCH C01 INTO :TM_DB2_DATABASE,:TM_DB2_TABLESPACE,
       :TM_DB2_REASON_CODE,:TM_DB2_OBJ_TYPE;
END;
IF SQLCODE < 0 THEN
  CALL SQL_ERROR;
PUT SKIP
  EDIT ('TOTAL FETCH COUNT:',WS_C2)
   (A,X(2),A);
PUT SKIP
  EDIT ('TOTAL UPDATE COUNT:',WS_C1)
   (A,X(2),A);
RETURN;

******************************************************************************
/* CHECK_DB_TS PROCEDURE */
/* MOST TM_DB2_DATABASE AND TM_DB2_TABLESPACE VALUES ARE REAL NAME. */
/* SOME OF THEM ARE OBID,PSID,DBID,ISOBID. THIS PROC CHECKS THIS */
/* VALUE AND SEARCHES IN CATALOG TABLES TO FIND REAL NAME. */
******************************************************************************

CHECK_DB_TS: PROC;
  WS_BINDB = $;
  WS_BINTS = $;
  WS_DB_NAME = TM_DB2_DATABASE;
  WS_TS_NAME = TM_DB2_TABLESPACE;
  WS_DB_FLAG = ' ';
  WS_TS_FLAG = ' ';
  WS_RCODE = ' ';
  IF TM_DB2_DATABASE > 'Ø' THEN
    DO;
      WS_PIC1 = TM_DB2_DATABASE ;
      WS_DEC1 = WS_PIC1;
      WS_BINDB = WS_DEC1;
    END;
  IF TM_DB2_TABLESPACE > 'Ø' THEN
    DO;
      WS_PIC1 = TM_DB2_TABLESPACE;
      WS_DEC1 = WS_PIC1;
      WS_BINTS = WS_DEC1;
    END;
  IF WS_BINDB > Ø THEN
    DO;
      EXEC SQL
      SELECT NAME
       INTO :DB_NAME
      FROM SYSSIBM.SYSDATABASE
      WHERE DBID = :WS_BINDB;
      IF SQLCODE < Ø THEN
        CALL SQL_ERROR;
      IF SQLCODE = Ø THEN
        DO;
          WS_DB_NAME = DB_NAME;
          WS_DB_FLAG = '1';
          WS_RCODE = '1';
        END;
      END;
    IF WS_BINTS > Ø THEN
      DO;

EXEC SQL
SELECT NAME
INTO :TS_NAME
FROM SYSIBM.SYSTABLESPACE
WHERE DBNAME = :WS_DB_NAME
AND PSID = :WS_BINTS;
IF SQLCODE < 0 THEN
  CALL SQL_ERROR;
ELSE DO;
  WS_TS_NAME = TS_NAME;
  WS_TS_FLAG = '1';
  WS_RCODE = '1';
END;
ELSE DO;
  EXEC SQL
  SELECT NAME
  INTO :TB_NAME
  FROM SYSIBM.SYSTABLES
  WHERE DBNAME = :WS_DB_NAME
  AND OBID = :WS_BINTS;
  IF SQLCODE < 0 THEN
    CALL SQL_ERROR;
  ELSE DO;
    WS_TS_NAME = TS_NAME;
    WS_TS_FLAG = '1';
    WS_RCODE = '1';
  END;
END;
ELSE DO;
  EXEC SQL
  SELECT B.TSNAME
  INTO :TS_NAME
  FROM SYSIBM.SYSEX Indexes A,
  SYSIBM.SYSTABLES B
  WHERE A.DBNAME = :WS_DB_NAME
  AND A.ISOBID = :WS_BINTS
  AND A.TBCREATOR = B.CREATOR
  AND A.TBNAME = B.NAME;
  IF SQLCODE < 0 THEN
    CALL SQL_ERROR;
  ELSE DO;
    WS_TS_NAME = TS_NAME;
    WS_TS_FLAG = '1';
    WS_RCODE = '1';
  END;
END;
END;
END CHECK_DB_TS;
UPDATE TABLE: PROC;
  EXEC SQL
    UPDATE PDBA.TIMEOUT_REPORT1
    SET TM_DB2_DATABASE = :WS_DB_NAME,
        TM_DB2_TABLESPACE = :WS_TS_NAME
    WHERE CURRENT OF C01;
    IF SQLCODE < 0 THEN
      CALL SQL_ERROR;
      WS_C_COUNT = WS_C_COUNT + 1;
      WS_C1 = WS_C1 + 1;
      IF WS_C_COUNT = 100 THEN
        DO;
          EXEC SQL
            COMMIT;
            WS_C_COUNT = 0;
          END;
      END;
  END UPDATE_TABLE;
/* UPDATE TABLE */

SQL_ERROR: PROC;
  WS_SQLCODE = SQLCODE;
  EXEC SQL
    ROLLBACK;
    PUT SKIP
      EDIT ('SQLCODE ',WS_SQLCODE,'COMMIT COUNT',WS_C_COUNT)
        ( A,A,A,A )
      ;
      STOP
    ;
  END SQL_ERROR;
/* SQL_ERROR */

END DBAB101;

DDLS OF PDBA.TIMEOUT_REPORT1 TABLE

CREATE TABLESPACE PDBAP011 IN PDBAD001
  USING STOGROUP PDBAS000
    PRIQTY 13200 SECOQTY 6600
    FREEPAGE 0 PCTFREE 5
    GBPCACHE CHANGED
    BUFFERPOOL BP2
    LOCKSIZE ANY
    CLOSE YES
  ;
  COMMIT;
  CREATE TABLE PDBA.TIMEOUT_REPORT1
DB2LOGA – JCL FOR DB2 LOG ANALYZE AND TIMEOUT REPORT

//DB2LOGA JOB (ACCT), 'DB2-DB2LOGA',
// CLASS=A,RESTART=SQLI,MSGCLASS=X,MSGLEVEL=(1,1)

/*
/* DELETE SYSPDBA.SYS0.DBP1.LOG_TEMP DATASET **
/* DELETE SYSPDBA.SYS0.DBP2.LOG_TEMP DATASET **
*******************************************************************************/
//DELPS EXEC PGM=IEFBR14
//DELLOG1 DD DSN=SYSPDBA.SYS0.DBP1.LOG_TEMP,DISP=(MOD,DELETE,DELETE),
// SPACE=(CYL,(10,10))
//DELLOG2 DD DSN=SYSPDBA.SYS0.DBP2.LOG_TEMP,DISP=(MOD,DELETE,DELETE),
// SPACE=(CYL,(10,10))
/*
*******************************************************************************/
/* DEFINE SYSPDBA.SYS0.DBP1.LOG_TEMP DATASET **
/* DEFINE SYSPDBA.SYS0.DBP2.LOG_TEMP DATASET **
*******************************************************************************/
//DFPDS EXEC PGM=IEFBR14
//DFLOG1 DD DISP=(NEW,CATLG,DELETE),
// DSN=SYSPDBA.SYS0.DBP1.LOG_TEMP,
// SPACE=(CYL,(50,50)),DCB=(RECFM=FB,LRECL=132)
//DFLOG2 DD DISP=(NEW,CATLG,DELETE),
// DSN=SYSPDBA.SYS0.DBP2.LOG_TEMP,
// SPACE=(CYL,(50,50)),DCB=(RECFM=FB,LRECL=132)
/*
*******************************************************************************/
/* THIS STEP INCLUDES SDSF BATCH COMMANDS. **
/* LOAD DBP1MSTR LOG RECORD FROM SDSF TO DATASET **
/* LOAD DBP2MSTR LOG RECORD FROM SDSF TO DATASET **
*******************************************************************************/
//SDSGET EXEC PGM=ISFAFD
//ISFOUT DD SYSOUT=*  
//ISFIN DD *
SYSTYPE PX*
PREFIX DBP1MSTR
OWNER *
DA
FIND 'DBP1MSTR'
++S
PRINT ODSN 'SYSPDBA.SYS0.DBP1.LOG_TEMP' SHR
PRINT 1 999999
PRINT CLOSE
PREFIX DBP2MSTR
OWNER *
DA
FIND 'DBP2MSTR'
++S
PRINT ODSN 'SYSPDBA.SYS0.DBP2.LOG_TEMP' SHR
PRINT 1 999999
PRINT CLOSE
/*
*******************************************************************************/
/* RUN LOG ANALYSE REXX PROGRAM FOR DBP1 **
*******************************************************************************/
//DB2DP1  EXEC  PGM=IKJEFT01,DYNAMNB=30,REGION=4096K
//STEPLIB  DD  DSN=ISP.SISPLOAD,DISP=SHR
//SYSEXEC  DD  DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*  
//SYSTSIN  DD  *
PROFILE  NOPREFIX
%DB2LOGA  DBP1
/

/**************************************************************************
/*  RUN  LOG  ANALYSE  REXX  PROGRAM  FOR  DBP2  */
**************************************************************************/
//DB2DP2  EXEC  PGM=IKJEFT01,DYNAMNB=30,REGION=4096K
//STEPLIB  DD  DSN=ISP.SISPLOAD,DISP=SHR
//SYSEXEC  DD  DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*  
//SYSTSIN  DD  *
PROFILE  NOPREFIX
%DB2LOGA  DBP2
/

/**************************************************************************
/*  LOAD  TIMEOUT  RESULT  SET  TO  DB2  TABLE  */
**************************************************************************/
LOAD1  EXEC  DSNUPROC,PARM='DBP0,LOADTIM'
SORTLIB  DD  DISP=SHR,DSN=SYS1.SORTLIB
//SYSREC00  DD  DSN=SYSPDBA.PS0.DBP1.TIMEOUT,DISP=SHR
//    DD  DSN=SYSPDBA.PS0.DBP2.TIMEOUT,DISP=SHR
//SORTWK01  DD  SPACE=(CYL,(10,10),,,ROUND)
//SORTWK02  DD  SPACE=(CYL,(10,10),,,ROUND)
//SORTWK03  DD  SPACE=(CYL,(10,10),,,ROUND)
//SORTWK04  DD  SPACE=(CYL,(10,10),,,ROUND)
//SORTOUT  DD  SPACE=(CYL,(10,10),,,ROUND)
//SYSMAP  DD  SPACE=(CYL,(10,10),,,ROUND)
//SYSUT1  DD  SPACE=(CYL,(10,10),,,ROUND)
//SYSSIN  DD  *
LOAD  DATA
INDNN  SYSREC00
PREFORMAT
REPLACE
STATISTICS  TABLE  (ALL)
    INDEX  (ALL  KEYCARD  FREQVAL  NUMCOLS  1  COUNT  10)
REPORT  NO
UPDATE  ACCESSPATH
LOG  NO
NOCOPYPEND
SORTKEYS
INTO  TABLE
    PDBA.TIMEOUT_REPORT1
    (   
        TM_PLAN   POSITION(1   )
        CHAR(8),
        TM_PLAN_CORR_ID  POSITION(10  )
    )
CHAR(12),
TM_PLAN_CONN_ID
CHAR(8),
TM_DATE
DATE EXTERNAL(10),
TM_TIME
TIME EXTERNAL(8),
TM_PLAN HOLDER
CHAR(8),
TM_PLANHD CORR ID
CHAR(12),
TM_PLANHD_CONN_ID
CHAR(8),
TM_DB2_MEMBER
CHAR(4),
TM_DB2_DATABASE
CHAR(8),
TM_DB2_TABLESPACE
CHAR(8),
TM_MESSAGE_CODE
CHAR(8),
TM_DB2_REASON_CODE
CHAR(8),
TM_DB2_OBJ_TYPE
CHAR(8)

/*

******************************************************************************
** UPDATE DB2 TABLE TO FIND AND REPLACE  **
** PSID,DBID,OBID,ISOBID WITH DATABASE NAME AND TABLESPACE  **
** NAME  **
******************************************************************************

//STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=PSDN.SDSNEXIT
// DD DISP=SHR,DSN=PSDN.SDSNLOAD
//SYSTSPRT DD SYSOUT=*
//SYSSPRINT DD SYSOUT=*
//SYSSDUMP DD SYSOUT=*
//SYSTSN DD *
DSN SYSTEM(DBP0)
RUN PROGRAM(DBAB101) PLAN(DBAB101) -
 LIB('SYSPDBA.PD0.LOADLIB')
END

******************************************************************************
** SAMPLE TIMEOUT REPORTS FROM PDBA.TIMEOUT_REPORT TABLE  **
******************************************************************************

//SQL1 EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=PSDN.SDSNEXIT,DISP=SHR
// DD DSN=PSDN.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPUT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSTIN DD *  
DSN SYSTEM(DBP0)  
RUN PROGRAM(DSNTFZ) PLAN(DSNTFZ) -  
LIBRARY('PDSN.RUNLIB.LOAD')  
//SYSDUMP DD SYSOUT=*  
//SYST DD *  
SELECT A.TM_DATE,A.TM_PLAN,A.TM_PLAN HOLDER,  
       B.NAME AS TABLE_NAME,COUNT(*) AS TIMEOUT_COUNT  
FROM PDBA.TIMEOUT_REPORT1 A, SYSIBM.SYSTABLES B  
WHERE  
       A.TM MESSAGE_CODE = 'TIMEOUT'  
AND A.TM_DB2_DATABASE = 'DSNB06'  
AND B.DBNM = A.TM_DB2_DATABASE  
AND B.TSNM = A.TM_DB2_TABLESPACE  
AND B.TYPE = 'T'  
GROUP BY A.TM_DATE,A.TM_PLAN,A.TM_PLAN HOLDER,  
       B.NAME  
HAVING COUNT(*) > 1  
ORDER BY 5 DESC,1,2,3,4;  
/*  
**************************************************************************************  
/* SAMPLE DEADLOCK REPORTS FROM PDBA.TIMEOUT_REPORT TABLE  **  
**************************************************************************************  
/*SQL2 EXEC PGM=IKJEFT0,EXECNBR=20  
//STPLIB DD DSN=PDSN.SDSNEXIT,DISP=SHR  
// DD DSN=PDSN.SDSNLDR,DISP=SHR  
//SYSTSPRT DD SYSOUT=*  
//SYSPUT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSTIN DD *  
DSN SYSTEM(DBP0)  
RUN PROGRAM(DSNTFZ) PLAN(DSNTFZ) -  
LIBRARY('PDSN.RUNLIB.LOAD')  
//SYSDUMP DD SYSOUT=*  
//SYST DD *  
SELECT A.TM_DATE,A.TM_PLAN,A.TM_PLAN HOLDER,  
       B.NAME AS TABLE_NAME,COUNT(*) AS TIMEOUT_COUNT  
FROM PDBA.TIMEOUT_REPORT1 A, SYSIBM.SYSTABLES B  
WHERE  
       A.TM MESSAGE_CODE = 'DEADLOCK'  
AND A.TM_DB2_DATABASE = 'DSNB06'  
AND B.DBNM = A.TM_DB2_DATABASE  
AND B.TSNM = A.TM_DB2_TABLESPACE  
AND B.TYPE = 'T'  
GROUP BY A.TM_DATE,A.TM_PLAN,A.TM_PLAN HOLDER,  
       B.NAME  
HAVING COUNT(*) > 1  
ORDER BY 5 DESC,1,2,3,4;  
/*
CPLIDB2 – SAMPLE COMPILE-BIND PROCEDURE FOR PLI-DB2-BATCH PROGRAM

CPLIDB2:
//CPLIDB2 JOB , ,MSGLEVEL=(1,1),MSGCLASS=X,CLASS=A,
// NOTIFY=&SYSUID,REGION=0M

******************************************************************************
// VA PLI COMPILE JOB FOR DB2-BATCH-VAPLI
******************************************************************************
// ** PRECOMPILE, COMPIL, LINK-EDIT VE BIND PROCEDURE
******************************************************************************
// ----------------------------------------------

/* DB2 PRECOMPILE STEP */
// ----------------------------------------------

//DB2 EXEC PGM=DSNHPC,PARM='HOST(PLI),SOURCE,DEC(31)',REGION=2M
//DBRMLIB DD DSN=SYSPDBA,PD0,DBRMLIB(DBAB101),DISP=SHR
//STELIB DD DSN=PSMN.SDSNLOAD,DISP=SHR
// DD DSN=PSMN.SDSNEXIT,DISP=SHR
// DD DSN=VAPLI.SIBMZCMP,DISP=SHR
// DD DSN=CEE.SCEERUN,DISP=SHR
//SYSLIB DD DSN=SYSPDBA.PD0.SRCLIB,DISP=SHR
//SYSCIN DD DSN=&DSNHOUT,DISP=(MOD, PASS),
// UNIT=VIO,SPACE=(TRK,(20,10))
//SYSPRINT DD SYSOUT=X
//SYSTEMR DD SYSOUT=X
//SYSDUMP DD SYSOUT=X
//SYST1 DD UNIT=VIO,SPACE=(TRK,(20,10))
//SYST2 DD UNIT=VIO,SPACE=(TRK,(20,10))
//SYSIN DD DSN=SYSPDBA.PD0.SRCLIB(DBAB101),DISP=SHR

******************************************************************************
// VA PL/I COMPILE STEP
******************************************************************************

//PLI EXEC PGM=IBMZPLI,COND=(8,LT),REGION=2M,
// ** PARM=('AF,F(I),XREF(FULL),INC,M,NEST,OF,S,MAP,EXTRN(FULL)',
// 'PARM=('OBJECT,OPTIONS,INC,S,MAP,FLAG(I 250)',
// 'LIMITS(FIXEDDEC(31),FIXEDBIN(63)),OFFSET,GN')
//STELIB DD DSN=VAPLI.SIBMZCMP,DISP=SHR
// DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=X
//SYSLIN DD DSN=&OBJ,UNIT=VIO,DISP=(&(PASS),
// SPACE=(TRK,(20,10)),DCB=(RECFM=FB,LRECL=80,BLKSIDE=32000)
//SYST1 DD UNIT=VIO,SPACE=(TRK,(20,10))
//SYST2 DD UNIT=VIO,SPACE=(TRK,(20,10))
//SYSIN DD DSN=&DSNHOUT,DISP=(OLD,DELETE)

******************************************************************************
// PRE-LINK EDIT STEP
******************************************************************************

//preaded EXEC PGM=ECPRBLK
//STELIB DD DSN=CEE.SCEERUN,DISP=SHR
//SYSSMSG DD DSN=CEE.SCEEMSGP(ECPRMSGE),DISP=SHR
//SYSLIB DD DUMMY
//SYSMOD DD DSN=&&PLNK,DISP=(,&PASS),UNIT=VIO,SPACE=(CYL,(1,1)),


© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
SAMPLE TIMEOUT REPORT

+---------------------------------------------+
| TM_DATE | TM_PLAN | TM_PLAN HOLDER | TABLE_NAME | TIMEOUT_COUNT |
+---------------------------------------------+
| 2001-08-23 | HSP207 | HSPB151 | HSOSZLESME | 22 |
SAMPLE DEADLOCK REPORT

<table>
<thead>
<tr>
<th>TM_DATE</th>
<th>TM_PLAN</th>
<th>TM_PLAN HOLDER</th>
<th>TABLE_NAME</th>
<th>TIMEOUT_COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001-08-27</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSEMIR</td>
<td>10</td>
</tr>
<tr>
<td>2001-09-14</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>9</td>
</tr>
<tr>
<td>2001-08-17</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>7</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>7</td>
</tr>
<tr>
<td>2001-10-05</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>7</td>
</tr>
<tr>
<td>2001-10-10</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>7</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>MKLPNL20</td>
<td>MKLB429</td>
<td>MKFIZIKI</td>
<td>6</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>MKLPNL20</td>
<td>MKLB429</td>
<td>MKPRFYI</td>
<td>6</td>
</tr>
<tr>
<td>2001-09-13</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>6</td>
</tr>
<tr>
<td>2001-09-21</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>6</td>
</tr>
<tr>
<td>2001-10-08</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>6</td>
</tr>
<tr>
<td>2001-08-28</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>5</td>
</tr>
<tr>
<td>2001-09-07</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>5</td>
</tr>
<tr>
<td>2001-08-24</td>
<td>POSPLN01</td>
<td>OPOS032</td>
<td>KONTOR_DOSYASI</td>
<td>4</td>
</tr>
<tr>
<td>2001-08-27</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-14</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSEMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-17</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-19</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-25</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSEMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-26</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSEMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-09-28</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-10-01</td>
<td>CRBU001</td>
<td>CRBU001</td>
<td>CSTMR_AU_CR_ADAT</td>
<td>4</td>
</tr>
<tr>
<td>2001-10-03</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>4</td>
</tr>
<tr>
<td>2001-08-20</td>
<td>HSPSPL01</td>
<td>HSOLPL01</td>
<td>HSGRP_EMIR</td>
<td>3</td>
</tr>
<tr>
<td>2001-08-24</td>
<td>HSPSPL01</td>
<td>HSOLPL01</td>
<td>HSGRP_EMIR</td>
<td>3</td>
</tr>
<tr>
<td>2001-08-27</td>
<td>HSOLPL01</td>
<td>HSOLPL01</td>
<td>HSEMIR</td>
<td>3</td>
</tr>
<tr>
<td>2001-08-28</td>
<td>HSOLPL01</td>
<td>HSOLPL01</td>
<td>HSEMIR</td>
<td>3</td>
</tr>
<tr>
<td>2001-08-31</td>
<td>MKLBG004</td>
<td>MUHAPL01</td>
<td>INVESTMENT_AU</td>
<td>3</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>MKLPNL20</td>
<td>MKLPNL24</td>
<td>MKFIZIKI</td>
<td>3</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>MKLPNL20</td>
<td>MKLPNL24</td>
<td>MKPRFYI</td>
<td>3</td>
</tr>
<tr>
<td>2001-09-05</td>
<td>MKLPNL22</td>
<td>MKLB429</td>
<td>MKGUN_DETAY</td>
<td>3</td>
</tr>
<tr>
<td>2001-09-06</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSGRP_EMIR</td>
<td>3</td>
</tr>
<tr>
<td>2001-09-07</td>
<td>HSPSPL01</td>
<td>HSPSPL01</td>
<td>HSEMIR</td>
<td>3</td>
</tr>
</tbody>
</table>
IBM has beefed up its DB2 for Multiplatforms with a range of new and improved tools for AIX, HP-UX, Linux, Solaris, and Windows. Improved tools include DB2 Web Query Tool for Multiplatforms V1.2 and DB2 Table Editor for Multiplatforms V4.2.

The Table Editor accesses, updates, and deletes data across multiple DB2 database platforms and the Web Query Tool connects all users directly to multiple enterprise databases simultaneously, regardless of database size, hardware, operating system, or location. Both tools now have support for Informix Dynamic Server 9.x.

Among the three new tools is DB2 High Performance Unload, providing quick unload and extraction of data for movement across enterprise systems or for reorganization in-place.

DB2 Recovery Expert provides automated recovery, while DB2 Performance Expert consolidates, reports, analyses, and recommends changes on DB2 performance-related information.

For further information contact your local IBM representative.

* * *

IBM and Peregrine have announced plans to deliver infrastructure resource management applications to the public sector, telecommunications, and financial services industries.

Specifically, Peregrine’s AssetCenter, ServiceCenter, and Get-It software applications will be optimized for DB2 database software running on IBM eServer pSeries, xSeries, and zSeries machines.
Web address:

For further information contact:
Peregrine Systems, 3611 Valley Centre Drive, San Diego, CA 92130, USA.
Tel: (858) 481 5000.