98

DB2 update

December 2000

In this issue

DB2 Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues
A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

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

Editor
Trevor Eddolls

Contributions
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 contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Using TIMESTAMP columns versus DATE and TIME columns

A frequent requirement of applications is to store date and time information for a DB2 table. The date and time data may be required for any number of reasons, such as:

- To record the date and time a row was added to the table.
- To act as the primary key for a row.
- To store date and time information relevant to the data, such as when an account was opened, closed, or last modified.
- To record other temporal information about the data.

Of course, this list is not exhaustive. There could be many other reasons for wanting date and time data in DB2 tables. But the purpose of this article is really not to discuss why the date and time data is required, but to discuss the methods of storing this information in a DB2 table and the trade-offs for choosing one method over another.

DATE, TIME, AND TIMESTAMP

It is possible to store date and time data in a row of a DB2 table using two columns or one. The first method is to create one column of DATE data type and one column of TIME data type. For example:

```
CREATE TABLE T1
   (. . .
    NEW_DATE      DATE       NOT NULL WITH DEFAULT,
    NEW_TIME      TIME       NOT NULL WITH DEFAULT,
    . . .);
```

The second method is to use a single TIMESTAMP column. For example:

```
CREATE TABLE T1
   (. . .
    NEW_TS        TIMESTAMP  NOT NULL WITH DEFAULT,
    . . .);
```

Which is the better solution? Of course, the answer is, ‘it depends’.

Let's examine what it depends on.

KEEPING IT SIMPLE

With DATE and TIME you must use two columns. TIMESTAMP uses one column, which might simplify data modification. If the entire date and time is always changed each time the column is updated, then the TIMESTAMP solution should be easier to implement from a modification perspective. However, just the reverse may be true for data access or if the date and time ever need to be changed independently of one another.

If you need to access the time component separately from the date component, it will be necessary to use functions to break these components apart from one another in a TIMESTAMP column. For example:

```
SELECT DATE(NEW_TS)
FROM  T1;

SELECT TIME(NEW_TS)
FROM T1;
```

The DATE() and TIME() functions will return only the date and time components of a timestamp column or expression. This is a bit more complicated than simply selecting two columns that are already in the right format. And you will lose the microseconds component of the timestamp value if you convert it using the TIME() function.

STORAGE REQUIREMENTS

The combination of DATE and TIME columns requires seven bytes of storage. A DATE column will use four bytes and a TIME column will use three bytes of storage. A TIMESTAMP column always requires ten bytes of storage. This is true even though the external representation of a timestamp uses 26 bytes, the external representation of time data uses eight bytes, and the external representation of a date uses ten bytes.

So, using the combination of DATE and TIME columns will save space.

Storage requirements usually should not dictate database design issues, though. It is more important to ensure that the database design will satisfy the requirements of the application and users of the data. Therefore criteria such as precision and ease of use are usually more important than the amount of storage required. However, in certain circumstances when there are severe storage shortages at your shop, a saving of three bytes per row might be an influencing factor in your database implementations.

ACCURACY

The TIMESTAMP approach will provide greater time accuracy than the DATE and TIME approach. The DB2 TIMESTAMP data type stores time information to the microsecond level. The TIME data type provides accuracy only to the second. So, if precision is important, use TIMESTAMP. Use TIME if you want to ensure that the actual time is *not* stored down to the microsecond level.

DATE AND TIME ARITHMETIC

There are additional considerations if you need to perform date and time arithmetic on the columns. Date and time arithmetic may be easier to implement using TIMESTAMP data. This is the case because you might be able to get by with a single calculation using TIMESTAMP data versus a combination of DATE and TIME calculations that will then need to be combined.

Subtracting one TIMESTAMP from another results in a timestamp duration. A timestamp duration is a number represented in DECIMAL(20,6) format. It consists of a number of years, months, days, hours, minutes, seconds, and microseconds. The number will be expressed in the format ‘yyyyxxddhhmmsszzzzzz’, where yyyy is the number of years, xx is the number of months, dd is the number of days, hh is the number of hours, mm is the number of minutes, ss is the number of seconds, and zzzzzz is the number of microseconds. For example:

```
2000-07-31-10.30.15.610208 - 2000-06-20-09.25.08.508928 =
00000111010507.101280
```

The result indicates that there were 0000 years, 01 month, 11 days, 01 hour, 05 minutes, 07 seconds, and 101280 microseconds between the two timestamps.

To calculate a duration using DATE and TIME columns, two subtraction operations must occur – one for the DATE column and one for the TIME column. This will result in a date duration and a time duration that may need to be combined together (depending on the further processing required by the application for the date and time durations).

To be clear, let's define date and time durations. A date duration is a DECIMAL(8,0) that consists of a number of years, months, and days. It has of the format 'yyyymmdd', with yyyy representing the number of years, mm the number of months, and dd the number of days. A time duration is a DECIMAL(6,0) number with the format 'hhmmss', where hh represents the number of hours, mm the number of minutes, and ss the number of seconds.

FORMATTING

If formatting the date and time data is a major consideration for your application then DATE and TIME may be a better solution. DB2 provides formatting options for DATE and TIME columns via local DATE and TIME exits. This means that you can specify a DB2 subsystem-wide default for the way that date and time values should be displayed. Likewise the CHAR function and DATE and TIME precompiler options exist for formatting DATE and TIME data. If the date and time information is to be extracted and displayed on a report or by an on-line application, the availability of these DB2-provided facilities for DATE and TIME columns should be considered when making your decision.

These facilities are not available for TIMESTAMP columns. There is a single format for a timestamp value in DB2, and it looks like the sample shown previously. A timestamp value always begins with the date component, in the format YYYY-MM-DD combined using a hyphen to the time component in the format hh.mm.ss.zzzzzz.

Of course, you can use the DATE() and TIME() functions to separate the components of the timestamp value and then display them in the same way as DATE and TIME columns. Of course, once again, you will lose the microseconds component of the timestamp value when you convert it using the TIME() function.

PRIMARY KEY CONSIDERATIONS

Sometimes DB2 databases rely on timestamps to automatically generate primary key values. This is a useful tactic when there is no obvious primary key or when multiple column primary keys would be difficult to implement effectively.

To use timestamps for primary key values you can simply create a column and assign it the TIMESTAMP data type with the specification NOT NULL WITH DEFAULT. When inserting rows, do not provide a value for the column and DB2 will automatically assign the column a value equal to the current timestamp at the time the row is inserted. This approach can work well unless you are inserting new values so rapidly that two rows are inserted at exactly the same time (down to exactly the same microsecond). Be sure to provide retry logic in your application program in case duplicate key values are inserted. Check for SQLCODE –803, and, if received, simply retry the INSERT. It is possible to get duplicate key values even when the insert rate is low if two transactions try to insert to the table at exactly the same time.

The DATE and TIME approach does not work well for primary keys. Since TIME values provide accuracy only to seconds, the frequency of duplicate inserts would be much, much greater. Therefore, do not use the DATE and TIME approach if you are going to use the date/time values for primary keys. Use a TIMESTAMP column instead.

As of DB2 Version 7 (and, with a PTF, DB2 Version 6) you can use the new IDENTITY feature to automatically generate sequential values for primary keys. This approach is even better than the TIMESTAMP approach and should be used if it is available on your DB2 subsystem. It is better because DB2 will automatically generate a unique number and you will not need to worry about duplicates. It should also be more efficient than the TIMESTAMP approach.

SUMMARY

So there is no clear-cut answer as to which is better – TIMESTAMP or a combination of DATE and TIME columns. You will need to examine your application design requirements and modify your database design accordingly. Favour DATE and TIME if storage is at a premium, you only require time accuracy to seconds, and you have strict date and time formatting requirements. However, if your application requires complicated date and time arithmetic, time accuracy greater than seconds, or automatic timestamp primary key values, then favour the TIMESTAMP solution. If your application does not fall into either of these broad categories, then favour TIMESTAMP over the DATE and TIME solution because it is usually simpler to implement.

Craig S Mullins
Director, Technology Planning
BMC Software (USA)

Utility for generating recovery jobs using the REXX SQL interface

When you need to recover a database, it is often time-consuming to set up the jobs and also to fill in the correct dataset names, label names, etc. Despite the best efforts, it is always possible to omit some crucial parameter or to code inefficient JCL, which could result in the recovery jobs taking a longer time to recover. Recovery requests are always urgent, and it always helps to have a utility to generate the recovery jobs in a flash.

This utility uses the REXX SQL interface to access the information from the catalog tables and to build the JCL. The chief advantages of using REXX SQL instead of other approaches like the unload using DSNTIAUL are:

- Easy to set-up with no SYSPRINT, SYSREC00, SYSIN, or other datasets being required.
- Does not require code to parse the unloaded information.
- No special character processing is required.

- Provides cursor processing, which helps to write optimized code.
- The overall efficiency is better because the REXX SQL interface uses the call attach facility to interface to DB2 and to retrieve the data.

As with all utilities, it is still advisable to run this utility when there is no or very little use of the related tables.

Assumptions and operational considerations:

- Image copies are assumed to be done to a tape dataset.
- The utility will generate the JCL for recovery to image copies taken at either the tablespace or partition level.
- The recovery to a quiesce point is complicated by the possibility that the quiesce point could be at the tablespace level whereas the image copies could be at the partition level. DB2 will lay down the full image copy prior to the quiesce point requested and then apply the log. In this case, we need to specify recovery by partition (even though the quiesce is by tablespace).
- Recovery to a quiesce point can be easily accomplished without coding the image copy DD cards that DB2 will lay down. But that would result in inefficient processing and would require manual intervention for bypassing repeated mounts of the same tape. This utility uses a query to identify both the quiesce point and the prior image copies that DB2 will lay down for these special requirements. It will generate more efficient JCL as a result.

Features of the utility:

- It uses a panel for easy and user-friendly input data capture.
- It can generate recovery JCL for either TOCOPY or TORBA through the ‘F’ or ‘Q’ options respectively (see assumption above).
- The tablespaces to be recovered are given as a list in a sequential dataset. Each line in this dataset contains the database name (if all tablespaces in the database are to be recovered) or a database and a tablespace name separated by one or more blanks (if a specific

tablespace in a database is to be recovered). Blank lines will result in unexpected results.

- The recovery JCL is written into members named as either the database name (if only the database name is present) or the tablespace name (if database and tablespace names are provided).
- It always uses an ICTIME range to identify the SYSCOPY record to use for recovery. Even if more than one record is present in the ICTIME range specified, the record with the maximum TIMESTAMP value will be chosen by the utility. Hence it is imperative that the range is specified so that the required record has the maximum TIMESTAMP value.
- The output dataset will be deleted if it exists and a new one will be created. There will be a member for each line in the input dataset. It is assumed that the DD card limit of 1,635 per job will be sufficient for all cases.
- The image copies are assumed to be done to tape devices. Otherwise, relevant changes may be made to the SQL STMT used and the FILLJCL procedure.
- The SYSIN cards to REBUILD the INDEXES are written after the SYSIN cards for RECOVERY of all tablespaces in a database. This will ensure that the tape drive is released earlier for use by other jobs.
- Status messages are displayed even as the utility is executing.
- A member REPORT in the output dataset summarizes the input parameters and the status of the processing of each input record.

A pre-requisite is that the REXX SQL interface must be available for use. This was a free download for Version 5.0 for DB2 for OS/390. It is now available with Version 6.0.

Inputs to the utility are:

1 The DB2 subsystem ID to be used.

2 The dataset containing the list of database names or database and tablespacenames to be used for recovery.

3 The ICDATE for recovery.

4 An ICTIME range – a ‘From ICTIME’ and a ‘To ICTIME’ value.

5 The output dataset for writing the generated JCL. This will be allocated by the utility.

The output from the utility is one partitioned dataset with the given name and will have one member for each input record and a REPORT member as explained above.

This utility has two REXX programs and two panels:

- RECVER – the main program, which performs panel display validations and calls the program rcvrcal.
- RCVRCAL – the program called by the main program that performs all processing.
- RCVRPAN – the main data capture panel.
- RCVRERR – the panel for displaying error messages.

Site-specific information is marked in the code as SITESPEC. At our installation the production systems are on one machine and the test systems on another machine. You may need to modify the code in RECVER and RCVRCAL according to your site.

RCVRCAL

```
/* rexx */
/******************************************************************/
/* RCVRCAL   - by Jaiwant Jonathan                                */
/* Invocation: Invoked from RECVER                                */
/*  This REXX will generate the RECOVERY JCL for:                 */
/*    RECOVERY  TO AN IMAGE COPY                                  */
/*           OR TO A QUIESCE POINT                                */
/*  INPUT:  A  list of database names or database and tablespace  */
/*  names to be recovered as a sequential file.                   */
/*  Other inputs to be specifed are the DB2 subsystem ID,         */
/*  the ICDATE, the ICTIMEs between which the image copy          */
/*  or quiesce was done, and the name of an unallocated output    */
/*  dataset to write the results to.                              */
/*  Special Requirements:                                         */
/*  ------------------------                                      */
/*  This REXX uses the REXX SQL Interface facility provided by    */
/*  IBM. Please refer the IBM Web site on downloading and using   */
/*  this feature.                                                 */
/*  Site-specific information is marked as  *** SITESPEC ***       */
/*****************************************************************/
trace o
clear
PARSE UPPER ARG P_ssid P_dblst P_icdate P_ict1 P_ict2 P_out P_typ
/*     *** SITESPEC ***                    */
P_act = '123AB456'
sys.1 = "//SYSIN  DD  * "
address tso "ispexec  vget (zsysid)"
p_sysid=zsysid
/*  pass parameters to local values    */
sid = strip(P_ssid)
/*  say  P_ssid P_dblst P_icdate P_ict1 P_ict2 P_out P_typ   */
I_lstdsn = strip(P_dblst)
I_icdate = strip(P_icdate)
I_btime = strip(P_ict1)
I_etime = strip(P_ict2)
ods_name = strip(P_out)
P_typ    = strip(P_typ)
cd = date(U)
us_date = substr(cd,7,2)||substr(cd,1,2)||substr(cd,4,2)
Call P000_MAIN
exit(0)
/* Main routine */
P000_MAIN:
/* Allocate and read the input list of DBNAMEs and/or TSNAMEs */
Call GETDBLST
pref = strip(sysvar(syspref))
/* Delete output dataset name if it exists   */
xx = outtrap("zap.","*")
address tso "delete '"ods_name"'"
xx = outtrap("OFF")
/*  allocate the output dataset anew  */
address tso "alloc f(rpds) new unit(sysda) space(5,10)",
          "cyl dir(50) reuse dsname('"ods_name"')",
          "dsorg(po) blksize(3120) lrecl(80) recfm(f b)"
if  rc>0 then
do
   say  'Failed during allocation of 'ods_name
   pull temp
   exit(8)
end
say 'Results will be generated into...' ods_name
s = 0
osn = 0
/*  Connect to DB2  */
Call PERFCONN
/* Process each row in input dataset of DBNAMEs and/or TSNAMEs   */
do i = 1 to dbl.0
   Parse Var dbl.i dbname  tsname  rest
   I_dbname=strip(dbname);
   I_tsname=strip(tsname);
   Call PROCROW
end
Call CLEANUP
Call PERFDISC
say 'Doing cleanup...'
xx = outtrap("zap.","*")
address tso "free f(lstdd)"
address tso "FREE ddname(rpds)"
xx = outtrap("OFF")
say; say 'Completed processing..'
say 'Refer member REPORT in 'ods_name
return
/*  Allocate and read input dataset for DBNAMEs and/or TSNAMEs  */
GETDBLST:
"ALLOCATE DD(lstdd) DSN('"I_lstdsn"') REUSE SHR"
if  rc>Ø then
do
   say  'Failed during allocation of 'I_lstdsn
   pull temp
   exit(8)
end
"execio * DISKR lstdd (FINIS STEM dbl."
return
/* Form query using input row from flat file */
PROCROW:
if P_typ = 'F' then
do
STMT =" SELECT  DBNAME, TSNAME, DSNUM, FILESEQNO, "
STMT = STMT||" DSNAME, HEX(START_RBA)"
STMT=STMT||",ICDATE, ICTIME ",
      " FROM   SYSIBM.SYSCOPY A ",
      " WHERE DBNAME ='"I_dbname"'"
if I_tsname ¬= '' then
STMT = STMT||" AND  TSNAME = '"I_tsname"'"
STMT = STMT||" AND ICDATE = '"I_icdate"'",
      " AND ICTIME BETWEEN '"I_btime"' AND '"I_etime"'"
STMT = STMT||" AND ICTYPE = 'F' ",
             " AND ICUNIT = 'T'  "
STMT = STMT||" AND ICBACKUP = ' ' ",
      " AND TIMESTAMP = ( SELECT MAX(TIMESTAMP) FROM  ",
      " SYSIBM.SYSCOPY B ",
      " WHERE  A.DBNAME = B.DBNAME ",
      "   AND  A.TSNAME = B.TSNAME ",
      "   AND  A.DSNUM  = B.DSNUM  ",
      "   AND  B.ICDATE = '"I_icdate"'",
      "   AND  A.ICTYPE = B.ICTYPE ",
      "   AND  B.ICTIME BETWEEN '"I_btime"' AND '"I_etime"')",
      " ORDER BY 1,2,3,4,6,7  "
end
```

```
/* if  recovering to a quiesce point, then we need to identify the   */
/* previous image copies that DB2 will lay down and code the DD      */
/* cards for them; this query will identify the previous FULL image  */
/* copy dataset names and the START_RBA of the quiesce point required*/
if P_typ = 'Q' then
do
STMT =" SELECT A.DBNAME, A.TSNAME, B.DSNUM, B.FILESEQNO, "
STMT = STMT||" B.DSNAME, HEX(A.START_RBA)"
STMT=STMT||", A.ICDATE, A.ICTIME  ",
      " FROM   SYSIBM.SYSCOPY A, SYSIBM.SYSCOPY B",
      " WHERE A.DBNAME ='"I_dbname"'"
if I_tsname ¬= '' then
   STMT = STMT||" AND  A.TSNAME = '"I_tsname"'"
STMT = STMT||" AND A.ICDATE = '"I_icdate"'",
      " AND A.ICTIME BETWEEN '"I_btime"' AND '"I_etime"'"
STMT = STMT||" AND A.ICTYPE = 'Q' AND B.ICTYPE = 'F' "
STMT = STMT||" AND A.ICBACKUP = ' ' ",
      " AND A.DBNAME = B.DBNAME ",
      " AND A.TSNAME = B.TSNAME ",
      " AND A.TIMESTAMP = ( SELECT MAX(C.TIMESTAMP) FROM  ",
      " SYSIBM.SYSCOPY C ",
      " WHERE  A.DBNAME = C.DBNAME ",
      "   AND  A.TSNAME = C.TSNAME ",
      "   AND  A.DSNUM  = C.DSNUM  ",
      "   AND  C.ICDATE = '"I_icdate"'",
      "   AND  A.ICTYPE = C.ICTYPE ",
      "   AND  C.DBNAME ='"I_dbname"'"
if I_tsname ¬= '' then
   STMT = STMT||" AND  C.TSNAME = '"I_tsname"'"
STMT=STMT||"   AND  C.ICTIME BETWEEN '"I_btime"' AND '"I_etime"')",
      " AND B.TIMESTAMP = ( SELECT MAX(D.TIMESTAMP) FROM  ",
      " SYSIBM.SYSCOPY D ",
      " WHERE  D.DBNAME = B.DBNAME ",
      "   AND  D.TSNAME = B.TSNAME ",
      "   AND  D.DSNUM  = B.DSNUM  ",
      "   AND  D.DBNAME ='"I_dbname"'"
if I_tsname ¬= '' then
   STMT = STMT||" AND  D.TSNAME = '"I_tsname"'"
STMT=STMT||"   AND (( D.TIMESTAMP < A.TIMESTAMP ",
      "   AND  D.ICDATE =  A.ICDATE) OR D.ICDATE < A.ICDATE)",
      "   AND  D.ICTYPE = 'F') ",
      " ORDER BY 1,2,3,4,6,7  "
end
ERRFLG = Ø
```

Editor's note: this article will be concluded in the next issue.

Jaiwant K Jonathan
DB2 DBA
QSS Inc (USA)

Extracting from LISTCAT output – revisited

I have added two new functions to the REXX program that was published in *DB2 Update* Issue 78, April 1999.

The original program runs on VSAM datasets that are active in DASD. It did not cover datasets that are SMS-migrated. As a result the total space information in the report isn't correct if some of the tablespaces and indexes have not been used in days and been migrated to a tape or cheaper storage. The new program will find all of the migrated datasets in the database and will submit JCL to recall them before processing LISTCAT to extract dataset information such as volume ID, extents, percent utilization, and so on.

Secondly, in DB2 Version 6 a SQL environment is supported in a REXX program. This new feature allows me to use SQL in the program to access the DB2 catalog to get the PQTY and SQTY information of a tablespace or an index. Obtaining that information I updated the program to generate JCL that will alter the space allocation of a tablespace or an indexspace that has an excessive number of extents. A threshold value for the number of extents will be specified to the program as a parameter to choose tablespaces or indexes to generate the SQL ALTER statements in the JCL.

```
/************************** REXX **********************************/
/*  This program extracts dataset information from an IBM listcat */
/*  output for DB2 tablespaces or indexes such as                 */
/*     - Volume serial                                            */
/*     - Number of partitions                                     */
/*     - Number of extents                                        */
/*     - Space allocation                                         */
/*     - Total space of the database                              */
/*  In addition it will provide the following functions:          */
/*     1. Recall HSM migrated datasets.                           */
/*     2. Generate a job to alter the space allocation to reduce  */
/*        the number of extents and multi-volume status.          */
/*                                                                */
/*  Input parm:                                                   */
/*     listclvl - HLQ of the datasets to process                  */
/*                (Eg vcat.dsndbc.dbname)                         */
/*     ssn      - DB2 subsystem name                              */
/*     extlim   - Number of extents of a tablespace or index      */
/*                that will be in the JCL to alter PQTY, SQTY.    */
/*                Multi-volume datasets will always be included.  */
/*                                                                */
/*  Output:                                                       */
/*    Report in a SYSPRINT                                        */
/*    Similar report in a dataset to sort the lines by the volid. */
/*                                                                */
/*  Subroutines:                                                  */
/*    check_migration - Look for migrated datasets                */
/*    recaljcl  - Generate JCL to recall the migrated datasets    */
/*    subhreca  - Submit a HSM RECALL job                         */
/*    reduce_extents - Create a job to reduce the multiple extents*/
/*                                                                */
/******************************************************************/
/*     uid.LISTCAT  RECFM=VB,LRECL=133,BLKSIZE=13700              */
/*     uid.LCATTEMP RECFM=VB,LRECL=133,BLKSIZE=13700              */
/*    - Report migrated datasets                                  */
/*      The LISTC command with data parm does not show migrated   */
/*      datasets resulting in incorrect calculation of total space*/
/*      allocation and usage percentage.                          */
/*      Actually the migrated are listed with the option, ALL of  */
/*      the LISTC command and denoted as NONVSAM.                 */
/******************************************************************/

Arg listclvl ssn extlim
Parse var listclvl word1 '.' word2 '.' dbname
uid = userid()
jobcard2 = '//    CLASS=K,MSGCLASS=X,NOTIFY='uid

Address tso
"ALLOC DDNAME(listcdd) DSNAME('uid.listcat') SHR REUSE"
"ALLOC DDNAME(outdd) DSNAME('uid.lcattemp') SHR REUSE"
"ALLOC DDNAME(sqldd) DSNAME('uid.db2.cntl(altqty)') SHR REUSE"

/**********************************************************/
/*        Recall migrated datatsets if any                */
/**********************************************************/
Call check_migration
If Number_of_migrated > 0 then do
   Say 'Following' Number_of_migrated 'datasets have been migrated:'
   Do j = 1 to Number_of_migrated
      Say '    ' dsname.j
   End
   Call recaljcl
   Call subhreca
   Say 'Recall has been submitted. It will be a while for recall ',
       'processing completes.'
   /* *****   Try following to wait until recall finishes ***
   Do until Number_of_migrated = 0
      Say 'Wait to press the Enter key or type NOWAIT and hit Enter'
      Pull wait_recall
      If wait_recall ¬= 'NOWAIT' then call check_migration
      Else if wait_recall = 'NOWAIT' then signal Nowait
   End
   Say 'Recall processing has been completed..'
   *************************************************************/
End
Else say 'There are no migrated datasets with the HLQ' listclvl
Nowait:

/**********************************************************/
/*        Listcat process for the report                  */
/**********************************************************/
"EXECIO Ø DISKW listcdd (FINIS"
"LISTC LVL('"listclvl"') DATA ALLOC OUTFILE(listcdd)"
"EXECIO * DISKR listcdd (FINIS"
Numeric digits 7
number_of_output_lines = queued()
If number_of_output_lines <> Ø then do
   Say ''
   Say ''
   Say '                              Volume listing'
   Say '                                  as of  '
   Say '                              ' DATE()
   Say ''
   Say 'VCAT      ' 'TS/IX    ' 'PART  ' 'VOLUME ' '  EXT    ' 'TRKS ',
       'ALLOC PARM      ' '%USE'
   Say '------    ' '------   ' '----  ' '------ ' '  ---    ' '---- ',
       '----------      ' '---'
   End
Else exit

/*****************************************************************/
/*              Initialize variables                             */
/*****************************************************************/
vcat. = ''              /* VCAT name                             */
database_name. = ''     /* Database name                         */
tspace_name. = ''       /* Tablespace or indexspace name         */
partnum. = Ø            /* Partition number. Ø for nonparitioned  */
space_alloc_type. = ''  /* Space allocation in cylinder or track  */
primary_alloc. = Ø      /* Amount of primary alloc. in cyl or trk */
secondary_alloc. = Ø    /* Amount of secondary alloc in cyl/trk   */
space_parm. = ''        /* Value of the SPACE= parameter in JCL   */
volume_id. = ''         /* DISK volume id                         */
number_of_extents. = Ø    /* Number of extents of a tablespace    */
number_of_volumes. = Ø    /* Number of extents of a tablespace    */
high_alloc_rba. = Ø       /* Highest RBA of allocation            */
high_used_rba. = Ø        /* Highest RBA being used               */
OUTDD_rec. = ''           /* Stem variable to be written to OUTDD */
percent_of_usage. = Ø     /* Percent of usage of the object space */
```

```
number_of_used_tracks. = Ø    /* Number of tracks being used     */
number_of_alloc_tracks. = Ø   /* Number of tracks allocated      */
tsname_of2many_extents. = ''  /* Obj. name with too many extents */
db_of2many_extents. = ''      /* Database name where it belongs  */
partnum_of2many_extents. = Ø  /* Partition number of the object  */
total_of2many_extents. = Ø    /* Number of extents of the object */
total_extents_multivol_ds. = Ø /* Total extents for multi-vol ds */
number_of_datasets = Ø         /* Number of datasets in listcat  */
total_number_of_cyl_alloc = Ø  /* Total amount alloc. cylinders  */
total_number_of_trk_alloc = Ø  /* Total amount alloc. and tracks */
total_number_of_trk_used = Ø   /* Total amount tracks used       */
total_number_alloc_in_tracks = Ø /* Total allocaction in tracks  */
percent_of_usage_in_total = Ø    /* Total usage percentage       */
total_allocation_in_gbytes = Ø   /* Total allocation in gigabyte */
i = Ø                /*   Total number of datasets in the DB     */
j = Ø                /*                                          */
k = Ø                /*                                          */
m = Ø                /*   i + number of datasets in multi-volume */

Do number_of_output_lines

  /*-----------------------------------------------------------*/
  /* Pull from the second column skipping the control character */
  /*-----------------------------------------------------------*/
  Pull 2 word1 word2 word3 word4

  If word1 = 'DATA' then do
    /*---------------------------------------------------------*/
    /*  New dataset. Initialize number of volumes. Increment i. */
    /*---------------------------------------------------------*/
     i = i + 1
     number_of_volumes.i = Ø
     Parse var word3 part1 '.' rpart2
     vcat.i = part1
     Parse var rpart2 part2 '.' part3 '.' part4 '.' part5 '.' part6
     database_name.i = part3
     tspace_name.i = part4
     partnum.i = strip(substr(part6,2,3),l,'Ø')
     End

  If substr(word1,1,1Ø) = 'SPACE-TYPE' then do
    /*---------------------------------------------------------*/
    /*  Find the space allocation type and high allocation RBA  */
    /*---------------------------------------------------------*/
     Parse var word1 part1 '-' part2 '-' part3
     space_alloc_type.i = strip(part3,l,'-')
     If space_alloc_type.i = 'CYLINDER' then
        space_alloc_type.i = 'CYL'
     Else
        space_alloc_type.i = 'TRK'
```

```
   Parse var word2 part1 '-' part2 '-' part3 '-' part4
   high_alloc_rba.i = strip(part4,l,'-')
   End

If substr(word1,1,9) = 'SPACE-PRI' then do
  /*-------------------------------------------------------*/
  /*  Find the space allocation size and high used RBA       */
  /*  Calculate the usage by dividing hi-u-rba by hi-a-rba   */
  /*-------------------------------------------------------*/
   Parse var word1 part1 '-' part2 '-' part3
   primary_alloc.i = strip(part3,l,'-')
   If space_alloc_type.i = 'CYL' then
      total_number_of_cyl_alloc = ,
           total_number_of_cyl_alloc + primary_alloc.i
   Else
      total_number_of_trk_alloc = ,
           total_number_of_trk_alloc + primary_alloc.i
   Parse var word2 part1 '-' part2 '-' part3 '-' part4
   high_used_rba.i = strip(part4,l,'-')
   percent_of_usage.i = (high_used_rba.i * 100) % high_alloc_rba.i
   End

If substr(word1,1,9) = 'SPACE-SEC' then do
  /*-------------------------------------------------------*/
  /*  Find the space allocation size for secondary           */
  /*-------------------------------------------------------*/
   Parse var word1 part1 '-' part2 '-' part3
   secondary_alloc.i = STRIP(part3,l,'-')
   End

If substr(word1,1,6) = 'VOLSER' then do
  /*-------------------------------------------------------*/
  /*  Find the volumes where the dataset exists and number   */
  /*  of extents in each volume.                             */
  /*-------------------------------------------------------*/
   number_of_volumes.i = number_of_volumes.i + 1
   volume = number_of_volumes.i
   Parse var word1 part1 '-' part2
   volume_id.i.volume = strip(part2,l,'-')
   word4 = strip(strip(word4,l),t)
   Parse var word4 part1 '-' part2 '-' part3
   number_of_extents.i.volume = strip(part3,l,'-')
   number_of_alloc_tracks.i.volume = 0
   temp_count = number_of_extents.i.volume
   Do while temp_count > 0
     /*----------------------------------------------------*/
     /*  Calculate the space allocation for each volume      */
     /*----------------------------------------------------*/
      Pull 2 word1 word2 word3 word4
      If SUBSTR(word1,1,8) = 'LOW-CCHH' then do
```

```
          Parse var word3 part1 '-' part2
          number_of_alloc_tracks.i.volume = ,
             number_of_alloc_tracks.i.volume ,
                + strip(part2,l,'-')
          /*----------------------------------------------------*/
          /*  Decrement temp_count when LOW-CCHH card is read  */
          /*----------------------------------------------------*/
          temp_count = temp_count - 1
       End
    End
  End
End

/***************************************************************/
/*  Write the report                                           */
/***************************************************************/
Number_of_datasets = i
m = 1  /* Use for indexing OUTDD records */
l = 1  /* Index of records for datasets with too many extents */

Do j = 1 to Number_of_datasets
  /************************************************************/
  /* Format the variables before they are written to output   */
  /************************************************************/
  vcat.j = left(vcat.j,8)
  tspace_name.j = left(tspace_name.j,8)
  partnum.j = right(partnum.j,3)
  number_of_extents.j.1 = right(number_of_extents.j.1,3)
  number_of_alloc_tracks.j.1 = right(number_of_alloc_tracks.j.1,6)
  space_parm.j = space_alloc_type.j'('primary_alloc.j',' || ,
                 secondary_alloc.j')'
  space_parm.j = left(space_parm.j,13)
  percent_of_usage.j = right(percent_of_usage.j,3)

  /************************************************************/
  /* Write an output line.                                    */
  /************************************************************/
  Say vcat.j'   'tspace_name.j'   'partnum.j'   'volume_id.j.1'   ' ,
      number_of_extents.j.1'   'number_of_alloc_tracks.j.1'   ' ,
      space_parm.j'   'percent_of_usage.j

  /************************************************************/
  /* Accumulate for total amounts                             */
  /************************************************************/
  total_number_alloc_in_tracks = total_number_alloc_in_tracks ,
                               + number_of_alloc_tracks.j.1
  number_of_used_tracks.j.1 = number_of_alloc_tracks.j.1 ,
                           * percent_of_usage.j / 100
  total_number_of_trk_used = total_number_of_trk_used ,
                           + number_of_used_tracks.j.1
```

```
OUTDD_rec.m = vcat.j tspace_name.j partnum.j volume_id.j.1 ,
              number_of_extents.j.1 number_of_alloc_tracks.j.1 ,
              percent_of_usage.j

/*********************************************************/
/* Process multi-volume dataset                          */
/*********************************************************/
If number_of_volumes.j > 1 then do

   total_extents_multivol_ds.j = number_of_extents.j.1

   Do k = 2 to number_of_volumes.j
      /*************************************************/
      /*  Format the variables before write to output  */
      /*************************************************/
      number_of_alloc_tracks.j.k = ,
                right(number_of_alloc_tracks.j.k,6)
      number_of_extents.j.k = right(number_of_extents.j.k,3)
      total_extents_multivol_ds.j = total_extents_multivol_ds.j ,
                                 + number_of_extents.j.k

      /*************************************************************/
      /* Write a line for each volume for the same dataset.        */
      /* Multi-volume dataset will show percent usage in the line  */
      /* for the first volume, i.e the usage for the whole dataset */
      /*************************************************************/
      Say '                           ' volume_id.j.k ,
          '  ' number_of_extents.j.k ' ' number_of_alloc_tracks.j.k
      total_number_alloc_in_tracks = total_number_alloc_in_tracks ,
                                  + number_of_alloc_tracks.j.k
      number_of_used_trk.j.k = number_of_alloc_tracks.j.k ,
                             * percent_of_usage.j / 100
      total_number_of_trk_used = total_number_of_trk_used ,
                               + number_of_used_tracks.j.k
      m = m + 1
      OUTDD_rec.m = vcat.j tspace_name.j partnum.j volume_id.j.k ,
                   number_of_extents.j.k number_of_alloc_tracks.j.k
   End

   /*************************************************************/
   /* Find datasets with extents more than the limit provided   */
   /*************************************************************/
   If total_extents_multivol_ds.j > extlim then do
       db_of2many_extents.l = database_name.j
       tspace_of2many_extents.l = tspace_name.j
       partnum_of2many_extents.l = partnum.j
       total_of2many_extents.l = total_extents_multivol_ds.j
       l = l + 1
   End
   Else nop
```

```
  End
  m = m + 1
End

/*******************************************************************/
/*  Format and write the totals lines to the report.               */
/*  Change the formula if necessary:                               */
/*    1 cylinder = 15 tracks                                       */
/*    1 track = 56,664 kilo bytes                                  */
/*******************************************************************/
total_number_of_cyl_alloc = trunc(total_number_alloc_in_tracks / 15)
total_number_of_trk_alloc = total_number_alloc_in_tracks ,
                           - (total_number_of_cyl_alloc * 15)
total_allocation_in_gbytes = ,
    total_number_alloc_in_tracks * 56.664 / 1000000
total_allocation_in_gbytes = format(total_allocation_in_gbytes,6,1)
percent_of_usage_in_total = ,
   (total_number_of_trk_used / total_number_alloc_in_tracks) *100
percent_of_usage_in_total = format(percent_of_usage_in_total,4,1)
Say ' '
Say ' Total 3390 space =' total_number_of_cyl_alloc 'CYLS',
    total_number_of_trk_alloc 'TRKS  ' ,
    total_allocation_in_gbytes 'GB  ' ,
    percent_of_usage_in_total '% used.'

/*********************************************************/
/*  Write the OUTDD records to disk.                     */
/*********************************************************/
Address tso
"EXECIO * DISKW outdd (STEM OUTDD_rec. finis"
"FREE FI(listcdd)"
"FREE FI(outdd)"

/*********************************************************/
/*  Create a job to reduce the multiple extents          */
/*********************************************************/
jobcard1 = '//'uid"A JOB (000),'Alter PQTY',"
sysaff = '//*MAIN  CLASS='ssn
steplib = '//STEPLIB DD DSN=MVS.'ssn'.DSNLOAD,DISP=SHR'
Queue jobcard1
Queue jobcard2
Queue sysaff
Queue '//SPUFI     EXEC  PGM=IKJEFT01,DYNAMNBR=20'
Queue steplib
Queue '//SYSTSPRT    DD  SYSOUT=*'
Queue '//SYSTSIN     DD  *'
Queue '     DSN SYSTEM('ssn')'
Queue '     RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP2) - '
Queue "             LIB('MVS."ssn".DSNLOAD') "
Queue '     END '
Queue '//SYSOUT      DD    SYSOUT=*'
Queue '//SYSPRINT    DD    SYSOUT=*'
Queue '//SYSUDUMP    DD    SYSOUT=*'
Queue '//SYSIN       DD    *'
Address tso "SUBCOM DSNREXX"
If RC then
   s_rc = RXSUBCOM('ADD','DSNREXX','DSNREXX')
Address DSNREXX
"CONNECT "ssn
Do k = 1 to l
   /*********************************************************/
   /*  SQL query to the DB2 catalog to get the information    */
   /*  on space allocation - PQTY and SQTY                    */
   /*********************************************************/
   sqlquery = ,
   "SELECT PQTY, SQTY, 'T' FROM SYSIBM.SYSTABLEPART WHERE TSNAME='"||,
   tsname_of2many_extents.k || "' AND DBNAME = '" || ,
   db_of2many_extents.k || "' AND PARTITION = " ,
   partnum_of2many_extents.k ,
   "UNION ALL"
   "SELECT A.PQTY, A.SQTY, B.CREATOR FROM SYSIBM.SYSINDEXPART A," ,
                    "SYSIBM.SYSINDEXES B" ,
   "WHERE B.INDEXSPACE = '" || tsname_of2many_extents.k || ,
   "' AND B.DBNAME = '" || db_of2many_extents.k || ,
   "' AND B.NAME = A.IXNAME AND B.CREATOR = A.IXCREATOR" ,
   "AND A.PARTITION = " partnum_of2many_extents.k

   /*********************************************************/
   /*             Execute the query                          */
   /*********************************************************/
   "EXECSQL DECLARE C1 CURSOR FOR S1"
   "EXECSQL PREPARE S1 FROM :SQLQUERY"
   "EXECSQL OPEN C1"
   "EXECSQL FETCH C1 INTO :priqty, :secqty, :tori"

   /*****************************************************/
   /*   Calculate new priqty and secqty                  */
   /*****************************************************/
   new_priqty = (priqty + secqty * (total_of2many_extents.k -1)) * 4
   new_secqty = trunc(new_priqty / 10)
   /*********************************************************/
   /*      Create SQL ALTER to change the space allocation    */
   /*********************************************************/
   If tori = 'T' then do
      Queue ' ALTER  TABLESPACE 'db_of2many_extents.k'.' ||,
      Queue tsname_of2many_extents.k 'PART' partnum_of2many_extents.k,
      Queue 'PRIQTY 'new_priqty' SECQTY ' new_secqty ' ;'
      End
   Else do
      Queue ' ALTER  INDEXS 'tory'.' ||,
      Queue tsname_of2many_extents.k 'PART' partnum_of2many_extents.k,
      Queue 'PRIQTY 'new_priqty' SECQTY ' new_secqty ' ;'
      End
End
"DISCONNECT"

Address tso
"EXECIO * DISKW sqldd (finis "

Exit Ø

check_migration:
/**************************************************************/
/*          Look for migrated datasets                       */
/**************************************************************/
"LISTC LVL('"listclvl"') ALL OUTFILE(listcdd)"
"EXECIO * DISKR listcdd (FINIS"
number_of_output_lines = queued()
Number_of_migrated = Ø
j = Ø
vcat = ''
objname = ''
partnum = ''
Found_a_nonvsam = Ø
Do number_of_output_lines
  Pull 2 word1 word2 word3 word4  /* skip the 1st col. cc */
  If word1 = 'NONVSAM' then do
       Found_a_nonvsam = 1
       Parse var word3 part1 '.' rpart
       Parse var rpart part2 '.' part3 '.' part4 '.' part5 '.' part6
       End
  If substr(word1,1,6) = 'VOLSER' & Found_a_nonvsam = 1 then do
       Found_a_nonvsam = Ø
       Parse var word1 part7 '-' part8
       volume_id = strip(part8,l,'-')
       If volume_id = 'MIGRAT' then do /* This is a migrated dataset */
          Number_of_migrated = Number_of_migrated + 1
          vcat = part1               /* Save the vcat name */
          objname = part4            /* Save the obj name */
          partnum = substr(part6,2,3)
          dsname.Number_of_migrated =,
             vcat'.DSNDBC.'dbname'.'objname'.IØØØ1.A'partnum
       End
  End
End
Return

recaljcl:
/*********************************************************/
```

```
/*    Generate JCL to recall the migrated datasets     */
/*****************************************************/
jobcard1 = '//' || uid || "R JOB (ØØØ),'Recall DS',"
Queue jobcard1
Queue jobcard2
Queue '//HRECA  EXEC  PGM=IKJEFTØ1,DYNAMNBR=2Ø       '
Queue '//SYSTSPRT DD  SYSOUT=*                       '
Queue '//SYSPRINT DD  SYSOUT=*                       '
Queue '//SYSUDUMP DD  DUMMY                          '
Queue '//SYSTSIN  DD  *                              '
Do j = 1 to Number_of_migrated
  Queue " HRECA '" || dsname.j || "'"
End
Queue '/*                                            '
Queue '//*                                           '
Queue '                                              '
Return

subhreca:
/******************************************************/
/*    Submit a HSM RECALL job and notify the result     */
/******************************************************/
Address tso "SUBMIT *"
word3 = ''
Command_output. = ''
/* *********** Try this to check the RECALL output *****
x = outtrap('command_output.')
Do until word3 = 'ON OUTPUT QUEUE'
   "STATUS" uid'R'
   parse var command_output.1 word1 word2 word3
End
"OUTPUT" uid'R' "PRINT(*)"     /* Bring the job output in */
Num_of_output_lines = queued()
Do j = 2 to num_of_output_lines
   If substr(command_output.j,1,7) = 'IEF142I' then do
      Parse var command_output.j part1 ' COND CODE ' part2
      If part2 = 'ØØØØ' then
         say 'HRECAL has been submitted'
      Else say 'HRECAL can"t be submitted'
   End
End
 ************************************ */
Return
```

The JCL to execute the REXX program is the same as published in *DB2 Update*, Issue 78, April 1999.

Example output from the REXX program is shown below:

```
 //jobname  JOB (ØØØ),'SAM PARK',                                        *
 //     CLASS=K,MSGCLASS=X,MSGLEVEL=1,NOTIFY=uid,                        *
 //     USER=uid,PASSWORD=
 //STEP1    EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø
 //SYSTSPRT DD SYSOUT=*
 //SYSPRINT DD SYSOUT=*
 //SYSUDUMP DD DUMMY
 //AMSDUMP  DD DUMMY
 //SYSTSIN DD *
1READY
  EXEC 'uid.DB2.CNTL(LISTCAT11)' 'vcat.DSNDBD.ourdbt1 dsnx 5'
 Following 1ØØ3 datasets have been migrated:
      vcat.DSNDBC.ourdbt1.IXDYRVR.IØØØ1.AØØ1
      vcat.DSNDBC.ourdbt1.IXPRACC.IØØØ1.AØØ1
      vcat.DSNDBC.ourdbt1.IXPRACC.IØØØ1.AØØ2
      vcat.DSNDBC.ourdbt1.IXPRACC.IØØØ1.AØ31
      vcat.DSNDBC.ourdbt1.IXPRACC.IØØØ1.AØ32
        .................................
        .................................
        .................................
        .................................

 IKJ5625ØI JOB MYUIDØØR(JOB15776) SUBMITTED
1Recall has been submitted. It will be a while for recall processing
completes.

                                Volume listing
                                    as of
                                 13 Sep 2ØØØ

 VCAT     TS/IX     PART   VOLUME    EXT     TRKS  ALLOC PARM      %USE
 ------   ------    ----   ------    ---     ----  ----------       ---
 vcat     IXPZZLC      1   SGDD76      5      165    CYL(3,2)        9Ø
 vcat     IXPZZLC      2   SGDD51      5      165    CYL(3,2)        89
    .................................................................
 vcat     IX1RACE      1   SGDD7Ø      1    1Ø935    CYL(729,45)     99
 vcat     IX1RACE      2   SGDD63      1    1Ø935    CYL(729,45)     99
    .................................................................
 vcat     IX1RACR      1   SGDD73     49    2871Ø    CYL(15Ø,42)     98
                           SGDD76     19    1197Ø
    .................................................................
 vcat     IX1TRRR      1   SGDD78     82     1245    CYL(2,1)        99
    .................................................................

  Total 339Ø space = 37282 CYLS 1Ø TRKS       31.7 GB     89.2 % used.

 READY
 END
```

Samuel Park
DBA (USA)

Have you been cubed?

Fuzzy SELECT (see *Fuzzy Select*, *DB2 Update,* Issue 87, January 2000) can require GROUP BY to satisfy fuzzy queries. GROUP BY returns row sets in the result table for which the GROUPING expression is equal to column FUNCTIONs such as AVG, COUNT, MAX, MIN, and SUM. The following logical table is used for examples:

```
ALUMNI(ALUMNI_ID, surname, SCHOOL_ID, date, contribution)
```

GROUP BY was simple in the olden days:

```
      SELECT     school_id, SUM(contribution)
      FROM           alumni
      GROUP BY   school_id  --grouping expression
      ORDER BY   school_id  --order by is the way to guarantee row order
                            --in final group by result set
```

This returns a row set consisting of the aggregate alumni contribution within a school. An example result table is:

```
      school_id       SUM(contribution)
        Ø1              75ØØØ
        Ø2              9ØØØØ
        Ø3              85ØØØ
          …
```

A selection can be included (using WHERE):

```
      SELECT      school_id, SUM(contribution)
      FROM            alumni
      WHERE       contribution > 5ØØØ
      GROUP BY    school_id
      ORDER BY    school_id
```

This excludes any alumnus contribution below 5001 from processing by the GROUP BY.

The GROUP BY can have its own ‘WHERE’ (HAVING), which is applied to its intermediate result table:

```
      SELECT      school_id, SUM(contribution)
      FROM            alumni
      GROUP BY    school_id
      HAVING      SUM(contribution) > 84ØØØ      --drops school Ø1 above
      ORDER BY    school_id
```

WHERE and HAVING can be combined:

```
      SELECT      school_id, SUM(contribution)
      FROM            alumni
      WHERE       contribution > 5000
      GROUP BY    school_id
      HAVING      AVG(contribution) > 84000
      ORDER BY    school_id
```

This excludes any alumnus contribution below 5001 from being processed by GROUP BY and drops any school whose average contribution by selected alumnus is below 84001.

DB2 has added grouping-sets and super-groups to GROUP BY:

```
             ,<----------------------|
>>      GROUP BY  grouping-expression|                         ><
                  grouping-sets
                  super-groups


                   ,<----------------------|
>>      GROUPING SETS   grouping-expression|                   ><
                        super-groups
                              grouping-expression
                              super-groups
```

The grouping-expression is the original DB2 that identifies the grouping columns.

Restrictions on its use are:

- Column names must be unambiguous.
- Total length must be <= 254 bytes.
- It cannot include scalar-fullselect, variant function, or external action.

GROUPING SETS allow multiple grouping clauses in a single statement. It is logically equivalent to a UNION of two or more row groups in a single result table. GROUPING SETS can be a simple element, parenthesis-delimited element list, or super-group.

Super-group syntax looks like:

```
>>    ROLLUP       (grouping-expression-list)                  ><
      CUBE         (grouping-expression-list)
      grand-total  (-)
```

ROLLUP calculates the super-aggregate sub-total rows by applying the same column functions that are used to obtain regular rows.

CUBE returns a result table containing all the ROLLUP rows plus additional cross-tabulation rows where the GROUP BY syntax is identical.

The grouping-expression-list defines elements.

The grand-total is self-explanatory. ROLLUP and CUBE automatically compute a grand total so grand-total specification is unnecessary.

The following:

```
>>GROUPING      (expression)                        ><
```

is used to indicate whether a row specified by an expression is a regular row or a super-aggregate row. It returns 0 or 1, where 0 means a regular row and 1 indicates a super-aggregate NULL row usable for sub-totals.

Expanded GROUP BY provides additional flexibility and complexity, eg:

```
     SELECT      alumni_id, surname, school_id, YEAR(date) AS year,
                 AVG(contribution) AS contribution
     FROM            alumni
     WHERE       YEAR(date) = 1999
     GROUP BY    school_id
     ORDER BY    school_id, 5 DESC    --column 5 is avg (contribution)
```

This is a basic GROUP BY returning alumni contributions within a school by descending contribution for year 1999.

If you want to see the average and individual contributions:

```
     SELECT      alumni_id, surname, school_id, YEAR(date) AS year,
                 AVG(contribution) AS contribution
     FROM            alumni
     WHERE      YEAR(date) = 1999
     GROUP BY GROUPING SETS ((YEAR(date), alumni_id), (alumni_id, date))
     ORDER BY    school_id, 5 DESC
```

Assume the values in the ALUMNI table shown in Figure 1.

Then GROUPING SETS selects the result table shown in Figure 2.

The first two rows come from GROUPING SETS (YEAR(date),

alumni_id	surname	school_id	date	contribution
124	paul	3	121999	9500
124	paul	3	091999	9000
124	paul	3	061999	8500
632	vesely	9	121999	8500
632	vesely	9	091999	8000
632	vesely	9	061999	7500

Figure 1: ALUMNI table values

alumni_id	surname	school_id	year	date	contribution
124	paul	3	1999	-	9000 (AVG)
632	vesely	9	1999	-	8000 (AVG)
124	paul	3	-	121999	9500
124	paul	3	-	091999	9000
124	paul	3	-	061999	8500
632	vesely	9	-	121999	8500
632	vesely	9	-	091999	8000
632	vesely	9	-	061999	7500

Figure 2: GROUPING SETS result table

alumni_id) and the remaining rows from GROUPING SETS (alumni_id, date).

ROLLUP computes the grand-total, eg:

```
SELECT      alumni_id, surname, school_id, YEAR(date) AS year,
            SUM(contribution)
FROM            alumni
WHERE      YEAR(date) = 1999
GROUP BY ROLLUP    (YEAR(date), alumni_id)
ORDER BY    school_id, 5 DESC
```

This returns Figure 1 with the appended super-aggregate grand-total row of:

```
-    -    -    -    -    51000
```

The MONTH(date) can be inserted into the SELECT to better illustrate the use of CUBE:

```
SELECT      alumni_id, surname, school_id, YEAR(date) AS year
            MONTH(date) AS month,
            SUM(contribution) AS contribution
FROM            alumni
WHERE       YEAR(date) = 1999
GROUP BY CUBE      (YEAR(date), MONTH(date), alumni_id)
ORDER BY    school_id, 6 DESC
```

the CUBE result table is shown in Figure 3.

```
alumni_id    surname      school_id   year    month    contribution
124          paul         3           1999    12       9500
124          paul         3           1999    09       9000
124          paul         3           1999    06       8500
632          vesely       9           1999    12       8500
632          vesely       9           1999    09       8000
632          vesely       9           1999    06       7500
-            -            -           1999    12       18000
-            -            -           1999    09       17000
-            -            -           1999    06       16000
124          paul         3           1999    -        27000
632          vesely       9           1999    -        24000
-            -            -           -       -        51000
```

Figure 3: CUBE result table

The last row is the CUBE super-aggregate grand-total row. GROUPING operands allows an application program to determine which rows in Figure 3 are regular or super-aggregate. GROUPING returns 0 for all table rows without any (regular) and 1 for all table rows with any (super-aggregate).

An empty GROUPING SETS allows grand-total of selected rows plus standard rows.

```
SELECT      alumni_id, surname, MONTH(date) AS month, contribution
FROM            alumni
GROUP BY GROUPING SETS (alumni_id, MONTH(date), ())
ORDER BY    alumni_id
```

This returns Figure 4.

Many permutations of GROUP BY clauses are possible:

- Starting with the simplest:

```
alumni_id   surname    month   contribution
124         paul       12      9500
124         paul       09      9000
124         paul       06      8500
632         vesely     12      8500
632         vesely     09      8000
632         vesely     06      7500
-           -          -       51000
```

Figure 5: Empty GROUPING SETS results table

```
GROUP BY school_id
```

This is equivalent to:

```
GROUP BY GROUPING SETS((school_id))
```

-
  ```
  GROUP BY alumni_id, school_id, date
  ```

  This is equivalent to:

  ```
  GROUP BY GROUPING SETS((alumni_id, school_id, date))
  ```

- The ROLLUP grouping-expression order is significant:

  ```
  GROUP BY ROLLUP(alumni_id, school_id)
  ```

  This is equivalent to:

  ```
  GROUP BY GROUPING SETS((alumni_id, school_id)
                         (alumni_id)
                         ()      )
  ```

- Whereas:

  ```
  GROUP BY ROLLUP(school_id, alumni_id)
  ```

  is equivalent to:

  ```
  GROUP BY GROUPING SETS((school_id, alumni_id)
                         (school_id)
                         ()      )
  ```

- And:

  ```
  GROUP BY CUBE(alumni_id, school_id, date)
  ```

  is equivalent to:

```
GROUP BY GROUPING SETS((alumni_id, school_id, date)
                       (alumni_id, school_id)
                       (school_id, date)
                       (alumni_id)
                       (school_id)
                       (date)
                       ()     )
```

Using expanded GROUP BY allows sophisticated summary queries that can provide users with the needed answers quickly with the minimum of fuss. I recommend using a small test database to vet proposed GROUP BY clauses for correctness and accuracy.

My next article will illustrate using expanded GROUP BY in an application program for more flexibility and options.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

DRDA connectivity for DB2 UDB for OS/390 – part 2

This month we conclude our look at DRDA connectivity for DB2 UDB for OS/390.

COMMUNICATION MANAGER FOR DRDA

In all the options for connectivity that were discussed in part 1 of this article, the Communication Manager (CM) was an integral part of the configuration. Please revisit Figures 1, 2, and 3 in the previous article to see where the Communication Manager is situated.

So what is the Communication Manager? What is its purpose? And what does it do?

The Communication Manager is a software product from IBM that is designed to provide network management services for OS/2, AIX, Windows 95, and Windows NT. Communication Manager supports all sorts of communication protocols and device connection options. For those of us who are familiar with VTAM in the mainframe world, the Communication Manager can be looked at as the VTAM and NCP of the workstation. Communication Manager serves the same function as VTAM and NCP in MVS.

In order for DB2 Connect to communicate with DB2 UDB for OS/390 it needs the Communication Manager.

The previous sentence needs some qualification. It is only true if the protocol used was SNA. If DB2 on OS/390 was at Version 5.1 or above and the protocol used is TCP/IP, the Communication Manager is not needed in the configuration.

It goes without saying that even if one is at DB2 OS/390 Version 5.1 or above, one can still refuse to use TCP/IP and choose to use SNA. In that case one always needs the Communication Manager.

Several entries within the Communication Manager have to be configured to match and correspond to entries in the DB2 Connect Node Directory and to other entries in VTAM (assuming we are using SNA – please revisit the information previously discussed about the Node Directory for relevance).

Only one entry of the Communication Manager will be discussed here. The rest of the entries and their mapping to VTAM are shown in Figure 1 where you will see the whole picture of SNA mapping. (Compare Figure 1 with Figure 2, where you can see the whole picture if you are using TCP/IP protocol.)

This article is not a course on Communication Manager, so no details about CM will be discussed. However, some points relevant to understanding SNA connectivity will be discussed here.

In SNA, the Communication Manager acts as a Logical Unit (LU) partner in two and three-tier connectivity architectures. Its other LU partner is the DDF of DB2 for OS/390.

The Communication Manager takes information from DB2 Connect, packages it, and passes it to its DDF VTAM partner LU in the DB2 of OS/390.

Remember that one of the entries that was catalogued in the Node Directory of DB2 Connect was the Symbolic Destination Name. This

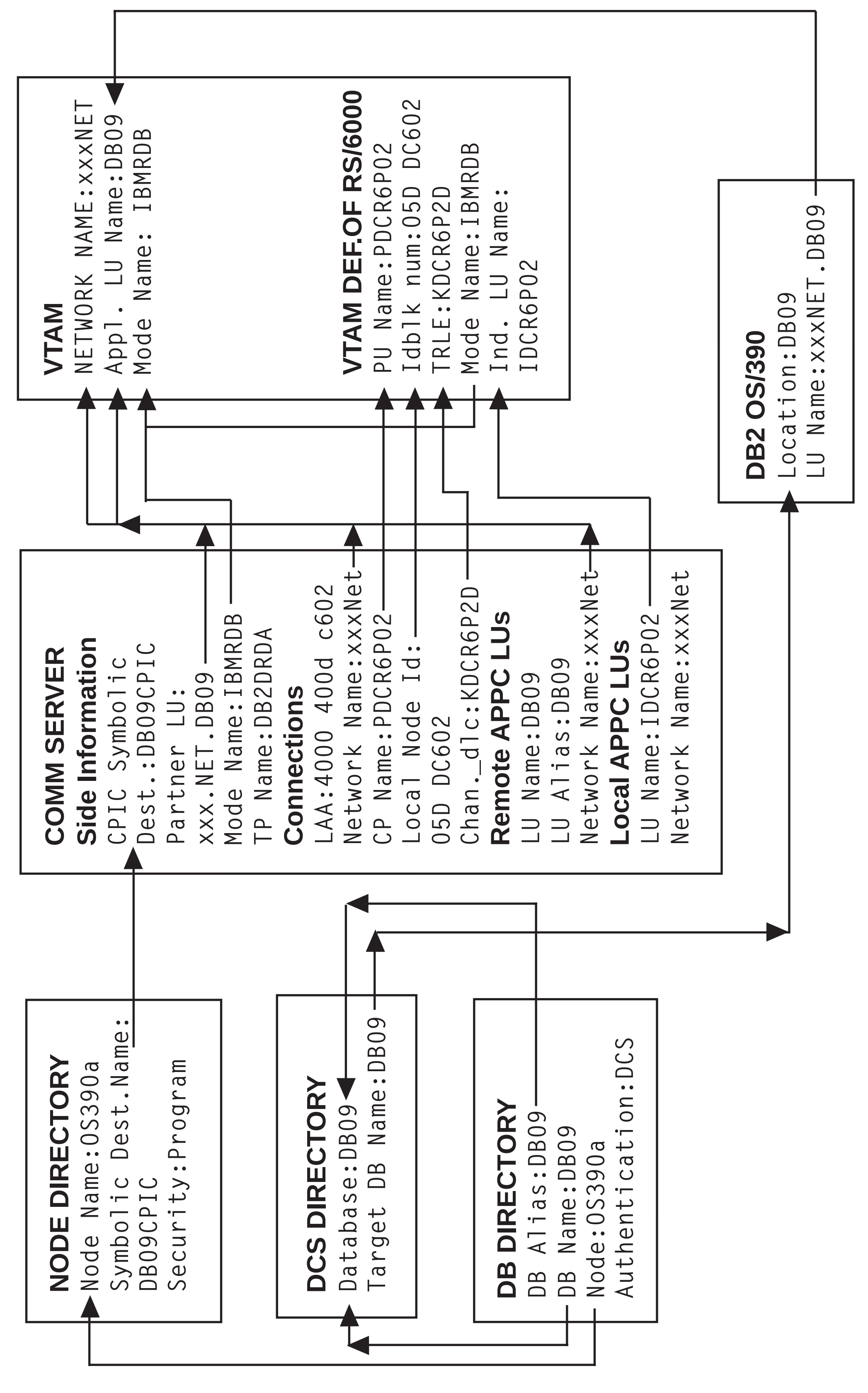

Figure 1: Connectivity

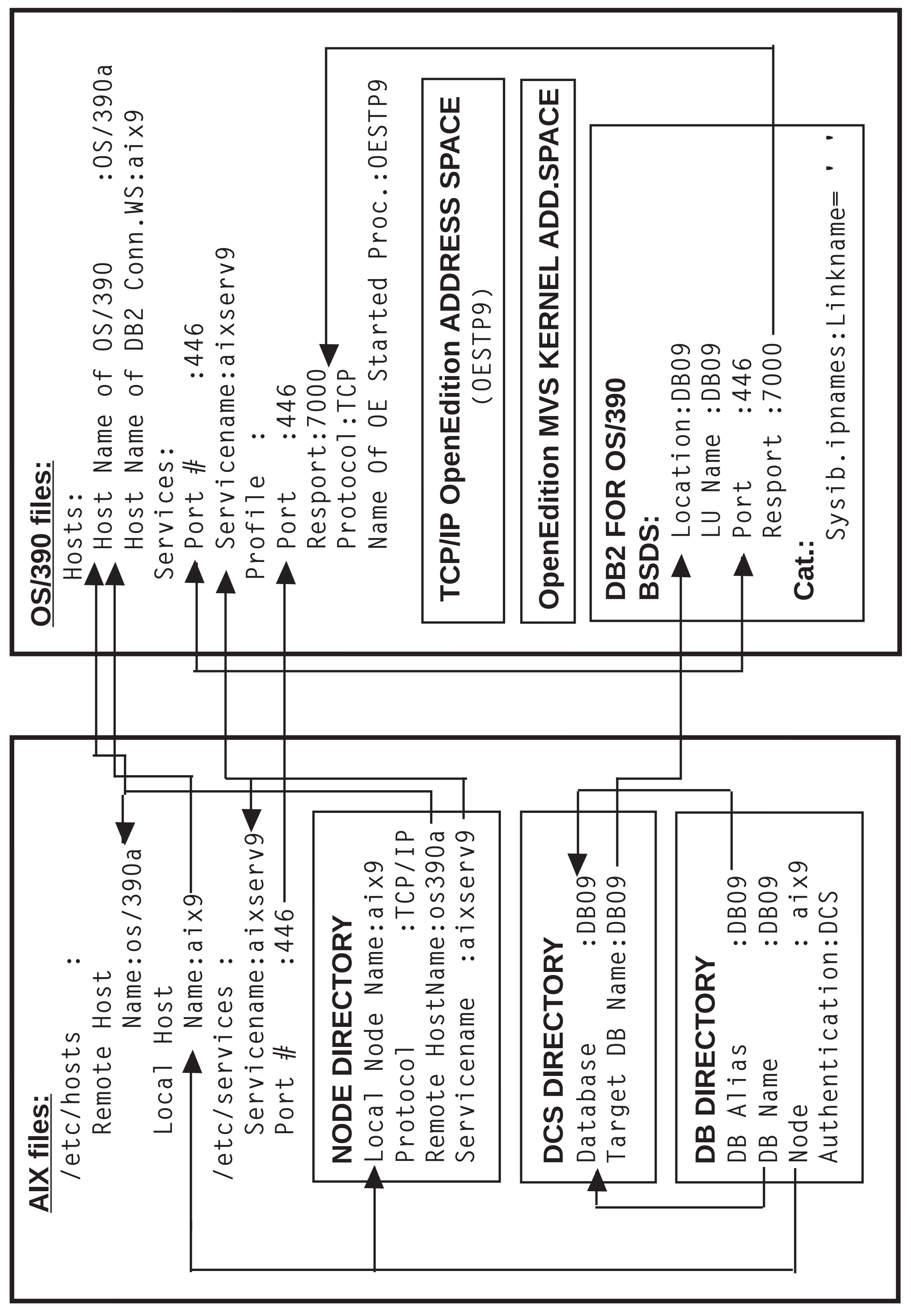

Figure 2: TCP/IP connectivity

entry must match in name and case the Side Information Name in the Communication Manager.

VTAM

For SNA connectivity, VTAM is an integral part of the configuration. One needs to configure VTAM with parameters that match the entries in the Communication Manager.

This article is not a course on VTAM. Nevertheless one should know the basic function of VTAM in the context of DRDA connectivity. VTAM needs to manage, carry, and facilitate network communications. In order to do that, VTAM needs to be aware of the remote and local LUs that are conversing, their PUs, the network they reside in, and their addresses. Please refer to Figure 1 .

VTAM also has to manage the communications of the DDF. The DDF is the only address space of DB2 that is a VTAM application and hence it needs a VTAM APPLID. Below is a sample VTAM definition for the DDF APPLID:

```
APDDCSD         VBUILD    TYPE  = APPL
DBØ9            APPL      APPC = YES                          X
AUTH=(ACQ),                                                   X
AUTOSES=1,                                                    X
DSESLIM=25Ø,                                                  X
EAS=9999,                                                     X
MODETAB=DB2MODES,                                             X
SECACPT=ALREADYV,                                             X
SRBEXIT=YES,                                                  X
VERIFY=NONE,                                                  X
DMINWNL=125,                                                  X
VPACING=1,                                                    X
SYNCLVL=SYNCPT,                                               X
DMINWNR=125,                                                  X
ATNLOSS=ALL
```

This is usually located in SYS1.VTAMLST(minor node name).

Please consult your DB2 manuals for the meaning of these important VTAM parameters used in the definition of the APPLID.

There is one entry that needs to be explained in the VTAM APPLID definition, namely the MODETAB. This entry, expanded below, shows the DRDA logmode, vertical and horizontal pacing, and the RUSIZE.

```
DB2MODES    MODETAB
IBMRDB      MODEENT     LOGMODE=IBMRDB,                         X
                        SSNDPAC=X'Ø1',                          X
                        SRCVPAC=X'ØØ',                          X
                        RUSIZES=X'8C8C',                        X
            MODEEND
            END
```

One can adopt these values verbatim because they were chosen with performance/efficiency in mind. This is true particularly for the value of the RUSIZE. The chosen value was selected to match the size of 32KB blocks that DB2 uses when it blocks the relational answer set using Limited Block Protocol, thus reducing the number of packets exchanged via the network.

The following formula explains how a hex value of X'8c8c' translates to 32KB. The 8 is the mantissa and C is the exponent. Here is the translation formula:

$8*2^{12} = 32768$ bytes

There is a point that was mentioned previously but is worth mentioning again here in the VTAM context. The configuration of VTAM is only needed with SNA connectivity. VTAM is not needed for TCP/IP connectivity. However VTAM is always needed for the DDF because the DDF is a VTAM application and needs an APPLID. LU6.2 (APPC) and TCP/IP are protocols that the DDF chose to use. At this point please compare Figures 1 and 2 to see the difference.

TCP/IP CONFIGURATION

TCP/IP consists of two parts, the Transmission Control Protocol (TCP) and the Internet Protocol (IP). Both parts constitute the Internet Protocol Suite. TCP/IP protocol has gained so much popularity lately that some experts believe it is going to become the preferred mode of communication even by IBM, if indeed it is not so already. One needs to look at the popularity of the Internet, Web-enabling applications, and e-commerce applications.

TCP/IP protocol has been mentioned several times in this article in relation to DRDA connectivity, whenever and wherever it was needed and usually in contrast to the LU6.2 protocol.

In this section of the article I need only to highlight the integrated DRDA TCP/IP connectivity as it is shown in Figure 2.

TCP/IP runs as a started task in its own address space which is sometimes referred to as a stack, engine, or TCP/IP instance. DDF of DB2 communicates with this stack using the OpenEdition MVS Kernel address space. Several DDFs (ie several DB2 subsystems) can communicate with one TCP/IP stack.

Figure 2 shows the various mappings from a DB2 Connect workstation situated in an RS/6000 machine with AIX as the operating system connecting to OS/390 and interacting with DB2 OS/390.

Please look at Figure 2. In the RS/6000 box there are two AIX files, namely /etc/hosts and /etc/services. These are very important files. The /etc/hosts file contains among other things the names of the local host and the remote host. The /etc/services file contains among other things the servicename and the port number that DB2 uses to listen on. This number is 446 and is the traditional port designated internationally for DRDA.

These two AIX files must have their equivalent datasets in the OS/390 system. The values in these two sets of files across the two operating systems should match or TCP/IP connectivity would not be established.

The values in the two AIX files mentioned above should also map to the values in DB2 Connect’s three directories. I have shown in previous pages the command needed to update DB2 Connect directories using TCP/IP protocol.

DB2 UDB FOR OS/390

So far all the components necessary for connectivity have been discussed with the exception of DB2 for OS/390. Now we turn our attention to DB2 OS/390.

For DB2 OS/390, the DDF should be defined (we already know that the DDF is a VTAM application and we have seen an example of its APPLID definition).

What is needed now is to show a sample of the Communication Record in the BSDS. A sample job and parameters for SNA Communication Record in the BSDS are shown below:

```
//DSNTLOG EXEC PGM=DSNJUØØ3,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=SYS1.DB2.V5R1MØ.SDSNLOAD
//SYSUT1   DD DISP=OLD,DSN=PXXØØØ.BSDSØ1
//SYSUT2   DD DISP=OLD,DSN=PXXØØØ.BSDSØ2
//SYSPRINT DD SYSOUT='*'
//SYSUDUMP DD SYSOUT='*'
//SYSIN    DD *
  DDF LOCATION=DBØ9,LUNAME=DBØ9,NOPASSWD
```

Below are the Communication Record parameters for TCP/IP connectivity:

```
//DSNTLOG EXEC PGM=DSNJUØØ3,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=SYS1.DB2.V5R1MØ.SDSNLOAD
//SYSUT1   DD DISP=OLD,DSN=PXXØØØ.BSDSØ1
//SYSUT2   DD DISP=OLD,DSN=PXXØØØ.BSDSØ2
//SYSPRINT DD SYSOUT='*'
//SYSUDUMP DD SYSOUT='*'
//SYSIN    DD *
  DDF LOCATION=DBØ9,LUNAME=DBØ9,NOPASSWD,
  RESPORT=7ØØØ,PORT=446
```

Please note the two ports shown above. These two DRDA ports allow the DDF to accept and send TCP/IP requests from any client that provides valid security information.

Port number 446 is the traditional DRDA port, and the other port (the resyncport) is used to process requests for two-phase commit resynchronization. This is needed so that applications which are precompiled with CONNECT 2 and TWO PHASE options can use the DB2 Connect over TCP/IP to update several DB2 OS/390s in one unit of work.

To enable such synchronization of all updates, one should update the DBM Config with the Transaction Manager value using the following command:

```
     Update dbm cfg using tm_database 1st_conn
```

It can be done using the Command Line Processor.

Note that these two ports are not owned by the DDF. They are owned by the TCP/IP OpenEdition address space but are used by the DDF.

There are two tables in the catalog of DB2 OS/390 that needs to be updated to enable connectivity. They are:

- For SNA, SYSIBM.SYSLUNAMES.
- For TCP/IP, SYSIBM.SYSIPNAMES.

In both of these catalog tables, a blank line entry would allow any incoming LU or TCP/IP machine to connect to DB2 OS/390. If this generalization is not acceptable to an organization, then the two tables should be updated with a specific LU name and IPNAME.

CONNECTIVITY PERFORMANCE

Three-tier connectivity performs better than two-tier connectivity because of the reduced traffic density of SQL statements coming from the workstation. Remember in this configuration the barrage of SQL statements is coming from the Application Server not from the workstation.

ESCON connectivity from the Application Server to the mainframe produces considerably better performance than T lines. I believe it can give you a speed of 17MB per second or more.

For SNA connectivity make sure to have:

- Vertical pacing set to 1 and RUSIZE in the MODTAB to 32KB.
- The delay parameters, NCP PCCU and LNCTL, on zero.

In \sqllib\DB2CLI.INI file (can be done transparently through CCA) make sure to have:

```
DefferedPrepare=1   /* this will compress prepare */
Cursorhold=0       /* this releases locks held by active idle threads */
Autocommit=0
RQRIOBLK=32767
```

In the DSNZPARM of DB2 OS/390 make sure to have:

```
CMTSTAT as inactive
IDTHTOIN=0
```

For heavy DRDA transactions, use the WLM, whether in a compatibility or goal mode, to adjust the dispatching priority of the DDF address space versus its transactions according to the concept of enclaves. With enclaves, each DDF transaction can be managed individually as a business unit of work and dispatched as a pre-emptive task rather than a local SRB.

For ERP applications that use dynamic SQL, enable the dynamic cache in the EDM pool of DB2 OS/390.

MOST COMMON CONNECTIVITY PROBLEMS

(Summarized from Connectivity Supplement of IBM's Product and Service Technical Library using the Internet.)

SQL0965 or SQL0969

Sometimes one may receive one of these two messages with various return codes. These are not specific messages – they mean that DB2 Connect could not understand and hence could not translate the SQLCODE returned by DB2.

Solution

Look in the MVS syslog or IOF of the DSNMSTR of DB2. Search for these messages and for clues in that area. Correct the problem and retry the transaction.

SQL1338

This means that the Symbolic Destination Name was not defined or defined inaccurately, or the case for the name did not match in the Node Directory and Side Information in the Communication Manager.

Solution

Revisit the discussion in this article about the integrated SNA mapping in Figure 2 and correct the problem.

SQL1403

This means that the user/password combination is not correct.

Solution

Revisit our discussion on updating the System Database Directory of DB2 Connect. Make sure that the authentication parameter is DCS or client. If the operating system is AIX, remember to include the password in the CONNECT statement.

SQL5043N

This means that there is a problem with the communication protocol either with LU6.2 or with TCP/IP

Solution

Revisit our discussion in DB2 Connect section and check that you have catalogued APPC or TCP/IP parameters properly.

Make sure that the environmental variable DB2COMM is defined correctly, such as DB2COMM=APPC,TCPIP.

Make sure that the entry in DB2COMM matches the entry in the Database Manager configuration.

Sometimes the host DB2 terminates a communication thread but the thread on the client for one reason or another is still active or hung. In this case you perform the following two steps:

- Issue the command 'db2 terminate' in order to explicitly terminate all CLP processes.
- Issue the command 'db2stop' in order to stop the current database instance.

SQL30020

This could mean there is an execution error at the host.

Solution

Consult the MVS syslog or IOF of the DSNMSTR of DB2 for ABEND codes and messages. Correct the problem accordingly.

SQL30060

This means that the client does not have the privilege to perform certain operations on the DB2 OS/390.

Solution

Make sure that the Communication Database (CDB) has the proper authentication entries in SYSIBM.SYSLUNAMES and SYSIBM.SYSIPNAMES.

SQL30061

This means that the DRDA location is wrong.

Solution

Revisit the configuration discussion of the DCS Directory in this article. The target entry in the DCS Directory should match the location name in the Communication Record in the bootstrap dataset (BSDS).

SQL30073 with RC=119C

This happens when the code page of DB2 OS/390 does not match the code page used by the CAE. Usually the code page of CAE is derived from the configuration of the operating system in which the CAE client is running.

Solution

If the OS of the client is Windows 95 do the following:

- Update the autoexec.bat file by inserting the following line:

```
DB2CODEPAGE=437
```

If the OS of the client is Windows NT do the following:

- Go to START/Control Panels/System Properties, choose the environment tab, and fill in the new variable DB2CODEPAGE and its value as 437.

SQL30081N with RC=1 and SNA sensecode=0877002C

This means that a wrong network name has been specified.

Solution

Revisit the integrated mapping between the Communication Manager and VTAM in Figure 2 of this article and correct the problem.

SQL30081N with RC=1 and SNA sensecode=ffff0003

This means the MAC address is not correct or the SNA link is not active.

Solution

Ask your VTAM professional to correct the problem.

SQL30081N with RC=1 and SNA sensecode=10030021

This signifies an LU name mismatch.

Solution

Revisit the integrated mapping between the CM and VTAM in Figure 2 in this article and correct the problem.

SQL30081N with RC=1 and SNA senscode=084B6031

This means that there was no remote thread allocation specified in your DSNZPARM.

Solution

Check the ‘maxbat’ value in your DSNZPARM and allocate enough threads.

Make sure that the SNA is up and running on the DB2 Connect workstation.

Make sure that the Distributed Data Facility (DDF) is up and running.

SQL30081N with RC=2 and sensecode=08120022

This means that the NCP and VTAM have a problem.

Solution

Verify that the parameter NUMILU of the NCP is not set to the default zero.

SQL30081N with RC=9 and sensecode=10086021

This means there is a mismatch in the TP.

Solution

Revisit our discussion of the integrated mapping between the CM and VTAM and correct the mismatch.

SQL30081N with RC=20

This means that a communication error has occurred.

Solution

Make sure that the SNA subsystem is up and running on the DB2 Connect workstation.

SQL30081N with RC=27 and sensecode=800Axxxx

This means the PIU name is wrong.

Solution

Check that the VTAM Path Information Unit (PIU) is correct.

SQL30081N with RC=79

This message is issued if a connection between DB2 Connect WS and the client failed.

Solution

Make sure the DB2COMM environmental variable states the protocol used as 'DB2COMM=TCPIP,APPC'.

Revisit the integrated mapping described earlier. A mismatch between the TCP/IP service name and/or port number may have occurred. Check the TCP/IP services files on both machines.

SQL30081N with protocol specific error code=10032

This message may be received when trying to disconnect from a machine where TCP/IP communication have already failed.

Solution

Restart the failed TCP/IP protocol service on the machine.

It may need recycling the entire machine.

SQL8002N

This means that the DB2 Connect licence has expired or was not installed properly.

Solution

Import the licence file available on the CD-ROM from this path:

```
    E:\DB2\LICENSE\DB2CONPR.LIC to your workstation
```

CONCLUSION

Various DRDA connectivity configurations from the workstation to DB2 OS/390 were discussed.

The role of various products that are involved in DRDA connectivity were also discussed in some detail.

A contrast was drawn in functionality and syntax between SNA and TCP/IP protocols whenever appropriate.

An integrated mapping between the various connecting components were shown in Figure 1 and 2.

Some common errors in connectivity that may be encountered were discussed and suggested solutions were given to these common errors.

Finally some thoughts were put forward that may result in better performance for DRDA applications through its various connectivity options.

Nicola S Nur
Senior DBA (Canada)

DB2 news

Tivoli has announced its Tivoli Business Systems Manager (TBSM), replacing Tivoli Manager for OS/390, while the Distributed Edition component replaces Tivoli Global Enterprise Manager (GEM).

Included is DB2 Performance Manager as a source of exceptions for DB2 resources, including tables and indexes, discovery and exception monitoring for IMS, and Tivoli Manager for OS/390 NT-based server.

The Distributed Edition can exploit Tivoli Distributed Monitors, including the ability to map generic distributed monitors to a software component, and supports CICS and DB2 instrumentation to monitor and control Distributed Edition applications on OS/390.

Functions include creation of lines of business via drag-and-drop, the ability to create LOB views containing both distributed and OS/390 resources, and a pre-packaged NetView application monitoring interface supported with DB2/CICS instrumentation.

For further information contact your local IBM representative.
URL: http://www.tivoli.com/news/press/pressreleases/en/2000/1005_tbsm.html.

* * *

EMC has announced a range of information sharing, movement, and back-up software, which supports Microsoft's Windows 2000 and .NET servers. The new EMC Extractor and ResourcePak for Windows promise enhanced functionality, increased access to data, and integration.

Extractor provides NT and Windows 2000 servers with direct non-disruptive access to information residing on mainframe-based DB2 databases at speeds up to seven times faster than traditional access methods, says the firm.

Based on OLE DB, it requires minimal mainframe resources. Once user authorization is provided by the mainframe, NT and Windows 2000-based applications directly access data residing on the EMC Symmetrix, enabling DB2 information sharing without disrupting the DB2 database.

For further information contact:
EMC Corp, 35 Parkwood Drive, Hopkinton, MA 01748, USA.
Tel: (508) 435 1000.
URL: http://www.emc.com/news/press_releases/view.jsp?id=521.

* * *

Quest Software has announced its Quest Central integrated suite of database management tools for DB2.

It promises to ensure availability, improve overall response time and performance, increase DBA productivity, and ease administration with core components, which include performance monitoring, SQL tuning, space management, and database administration.

For further information contact:
Quest Software, 8001 Irvine Center Drive, Irvine, CA 92618, USA.
Tel: (949) 754 8000.
Quest Software, The Priory, Stomp Road, Burnham, Bucks, SL1 7LS, UK.
Tel: (01628) 601000.
URL: http://www.quest.com/news/10-10-00_2.asp.