105 DB2 update

July 2001

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*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

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/contnote.html.

DB2 OLAP Server for OS/390 V7.1: DB2 Warehouse Manager

My first article on DB2 OLAP Server for OS/390 Version 7.1 appeared in Issue 102 of *DB2 Updat*e, April 2001. This article discusses the DB2 Warehouse Manager (DB2WM) component and UDB V7 performance enhancements.

IBM continues to omit the word 'Data' from its component description for reasons known only to them, although there are rumours the term 'data warehouse' has been copyrighted. DB2WM is a data warehouse manager because its objectives are to:

- Extract data from an operational environment.
- Allow access to a database maintaining data warehouse data.
- Provide user data.

IBM is trying to make things easier for DBAs by:

- Simplifying prototyping, development, and deployment of a data warehouse.
- Allowing the data centre to govern queries, analyse costs, manage resources, and track usage.
- Helping users find, understand, and access their data.
- Providing more flexible tools and techniques to build, manage, and access the users' data warehouse.
- Allowing IBM's expanded SQL to answer common user queries.

IBM has integrated Visual Warehouse Version 5.2 by automatically migrating it to the Control Center during UDB V7.1 install (see *Migrating to DB2 UDB Version 7.1 in a Visual Warehouse Environment*, SG24-6107, for details).

DB2WM can get data from lots of places including:

- Any DB2 family.

- Oracle, Sybase, Informix, and SQL Server (IBM is not willing to give up any database market particularly to Microsoft).
- Flat files (two-dimensional array).
- Data Joiner.
- And, for very old timers who thought IMS and VSAM were dead and buried, the Classic Connect Interface!

IBM enhanced Data Guide and renamed it Information Catalog. Its capabilities include:

- Helping users find, understand, and access their data by providing:
  - Data described in user-business terms.
  - User-friendly search engine.
  - User-friendly communication between user and content owner.
  - Access tool initiation.
- Information sharing your way by providing:
  - Support for any information object.
  - Support for almost anything including databases, queries, and Web pages.
  - Category grouping.
  - Correct object metadata.
  - Properties of almost anything.
  - Support for almost any user authority.
- Synchronization of information objects:
  - Automatic population with Data Warehouse Center.
  - Metadata interchange with the most popular access tools including QMF, DB2 OLAP, Brio, Business Objects, Cognos, Hyperion, etc.

IBM made it easy to connect to IC by providing:

- Windows (IBM does not want to but has no choice) or a Web browser (IBM hopes for Navigator).
- Almost any form of distribution including workgroups or a central repository.
- Storage by any DB2 family.
- Selected automatic population.

IBM supports bi-directional metadata interchange with Brio, Business Objects, and Cognos, allowing them to register their reports so users can launch them when they need those reports.

IBM provides you with a *free* DB2WM Agent daemon that listens to port name Verde and port number 11001. It has one agent process per command cycle using the DB2 CLI/ODBC interface for communication with the database. You also get an OS/390 Agent that runs under OS/390 Unix Systems Services to do:

- DB2 copying.
- Sample contents of a table or file.
- Execute user-defined programs.
- Use Data Joiner to access non-DB2 databases.
- Access IMS or VSAM data.
- Run Load, Reorg, Runstats, etc.
- Replication.

OS/390 Unix Systems Services lets Windows NT kernel request a DB2-to-DB2 data transfer using port 11001. OS/390 Daemon spawns a TCB using an ODBC **allocConnect** command for each request, thereafter using fetches and inserts for transfer. The TCB dies after the request is fulfilled. Windows NT accesses IMS or VSAM using a similar method.

DB2WM provides additional functionality to let data warehouses better satisfy their objective of providing users with timely and

accurate information for improving their bottom line. Additional UDB V7 performance and SQL enhancements allow DB2 star schema relational databases (see *Using a relational database for data warehouses*, *DB2 Update* Issue 83*)* to be increasingly effective data warehouses. Performance enhancements include increased parallelism for IN-list index access, more transform correlated subqueries, partition datasets parallel open, asynchronous INSERT preformatting, and improvements in ORDER BY and MIN/MAX.

DB2 V3 introduced parallelism with V6 extending it to IN-list access for a parallel group inner table (ACCESSTYPE=N). DBAs had to work hard to get this limited performance improvement. V7 allows parallelism on an IN-list predicate whenever it is supported by an index and parallelism is available. The table can be *inner*, *outer*, or *single*. EXPLAIN of PLAN_TABLE shows if parallelism is used.

DB2 can transform subqueries to a join of the subquery result table and outer query result table. This improves performance because DB2 can process joins as stage 1, whereas correlated subqueries are always stage 2. V7 has extended transformation to UPDATE, DELETE, and correlated subqueries using **IN**, **=ANY**, **=SOME**, and **EXISTS**. There are many restrictions so an EXPLAIN of PLAN_TABLE is required to determine whether a transform is occurring.

Opening and closing many datasets can result in many time-consuming tasks. V5 can use ten tasks whereas Version 6 can use twenty. V7 has twenty but can now do concurrent open and close partitions belonging to the same table and index space in parallel.

DB2 V1 to V4 synchronously preformatted new pages for INSERT if the search algorithm was exhausted, causing inconsistent execution times for INSERT-intensive applications. V5 added PREFORMAT option to the LOAD and RELOAD utilities that could be used if the final table space size was predictable and there was a high INSERT-to-read ratio. V7 asynchronously preformats allocated and not yet formatted but allocated pages, to ensure INSERT does not wait for page formatting.

UDB V7 can avoid sorting a WHERE clause with a **COL=constant** predicate by considering it as a constant in the result table and removing it from the ORDER BY list. V7 scrollable cursors let the Index Manager use a single index for MIN/MAX.

UDB V7 enhanced SQL features include UNION everywhere, scrollable cursors, expanded row expression for the IN subquery, and self-referencing UPDATE and DELETE. These features will be examined in future articles.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

Improving Query performance with correlated columns in DB2 V5

Many subscribers still use DB2 V5 and are looking for ways to enhance performance.

In DB2 V4, if a table has two or more correlated columns the optimizer may not choose an appropriate access path for queries having predicates on these columns. DB2 V5 supports collection of statistics on correlated columns in the catalog, allowing the optimizer to select the most efficient access path. This article explains the definition of correlated columns, problems related to access path selection in DB2 V4, and the new RUNSTATS options available in DB2 V5 as a solution to those problems.

WHAT ARE CORRELATED COLUMNS?

Two columns in a table are correlated if a value in one column is related to a value in another column in the same row. A very common example of correlated columns is in an Address table with columns CITY and STATE. These columns are correlated because if the value of column CITY in a row is 'LOS ANGELES', then the most likely value of column STATE in the same row is 'CA'. A table can have more than two columns as correlated columns. A column correlation can be either probable, possible, or definite, depending on the business rule and statistics collected on column values. If a business rule says that the value in one column is always dependent on the value in the other column (that is for all occurrences of the entity), then it is a

definite correlation. This may be the case when a table has been de-normalized from 3-NF to 2-NF. An example of this can be seen in an ORDER table containing columns SUPPLIER_ID and SUPPLIER_NAME. The correlation is probable when the relationship holds for a percentage of total number of rows. An example of this is EMPLOYEE table where 80% of women (SEX='F') are managers (GRADE='MGR'). The correlation is possible when there is no business rule, but the statistics collected on column values in a table suggest that there is a possible correlation between two columns.

DETECTING CORRELATED COLUMNS IN A TABLE

If you want to verify whether two columns, C1 and C2, in table T are correlated, issue Query1 and Query 2 and multiply the results. Now, issue Query 3. If the result of Query 3 is less than the product of the results of Query1 and Query 2, then columns C1 and C2 are most likely correlated columns.

Query 1:

```
SELECT  COUNT(DISTINCT C1)
FROM  T
```

Query 2:

```
SELECT  COUNT(DISTINCT C2)
FROM  T
```

Query 3:

```
SELECT  COUNT(*)
FROM  ( SELECT DISTINCT C1, C2
        FROM  T ) AS TEMP
```

PROBLEMS WITH CORRELATED COLUMNS IN DB2 V4

DB2 V4 does not provide any support for collecting and storing statistics on correlated columns in its catalog. Therefore, it assumes that the columns in a table are independent (except for full key columns of a composite index, where it stores the statistics in FULLKEYCARD column of table SYSIBM.SYSINDEXES). Consider Query 4 on TADDRESS table with correlated columns CITY and STATE. For this query, the optimizer thinks that columns

STATE and CITY are independent and therefore it applies a filter factor for both predicates without realizing that, once the predicate CITY='LOS ANGELES' has been applied on the rows, the second predicate STATE='CA' does not help any further in filtering (because all rows with CITY='LOS ANGELES' will also have STATE='CA'). Therefore, the optimizer underestimates the total number of result rows. But a person who has knowledge of the data model knows that the two columns are correlated and the actual number of qualifying rows may be more than the number estimated by the optimizer.

Query 4:

```
SELECT  *
FROM  TADDRESS
WHERE     CITY     =  'LOS ANGELES'
     AND  STATE    =  'CA'
```

ESTIMATING FILTER FACTOR WITH INDEX CARDINALITIES

Filter Factor (FF) is an estimate of the percentage of rows that will qualify according to a coded predicate. The DB2 optimizer uses a number of columns in catalog tables to calculate FF.

Consider an index, IADDRESS (CITY,STATE,ZIP), on TADDRESS table with CITY as the first key column, FIRSTKEYCARD equal to 100 and FULLKEYCARD equal to 10,000. The FIRSTKEYCARD column in table SYSIBM.SYSINDEXES contains a number of distinct values of the first column in an index and is the value used by the optimizer in calculating FF for a single column index. In our example, if the query has only a single predicate (CITY = 'LOS ANGELES'), then an FF of value 0.01 (1/FIRSTKEYCARD) and MATCHCOLS equal to 1 is used to decide the access path for a single predicate.

The FULLKEYCARD column in table SYSIBM.SYSINDEXES contains the number of distinct values of all bytes in a composite index and is used if an equal predicate is coded on each column in the composite index. In our example, if Query 4 has an additional predicate (ZIP='90014'), then an FF of value 0.0001 (1/FULLKEYCARD) and MATCHCOLS=3 is used to decide the access path for a compound predicate (all three ANDed predicates).

The COLCARD column in SYSIBM.SYSCOLUMNS table contains

the same value as the FIRSTKEYCARD for the first column of a composite index and is used if all columns of a composite index are not used with an equal predicate. COLCARD contains a statistical estimate of the number of distinct values in non-leading columns of a composite index (or a non-indexed column) if requested with RUNSTATS by specifying the table name and optionally the column names. If a matching index cannot be used on all columns in a composite index, COLCARD is used. If the beginning columns of a composite index are specified in the WHERE clause but the trailing columns are not specified, the optimizer uses FF as equal to 1/ COLCARD for each equal predicate on key column. For compound FF, the FFs for each of the ANDed predicates are multiplied, because the row must satisfy all the conditions. If the COLCARD for a column has not been updated, a default value of 0.04 is used. In our example of Query 4, the two equal predicates match only the first two key columns of the index IADDRESS, so the optimizer will use COLCARD for estimating FF. The COLCARD for column CITY is 100 (the same as FIRSTKEYCARD). If COLCARD for the column STATE is 50, then the FF for predicate (CITY = 'LOS ANGELES') is 0.01 (1/ COLCARD) and FF for the predicate (STATE = 'CA') is 0.02 (1/ COLCARD), and the estimated value for the compound FF is 0.0002 (0.01 * 0.02). For a compound FF of 0.0002 and MATCHCOLS equal to 2, the optimizer may decide to choose index for this query.

But, as we know, each value of CITY uniquely determines the value of STATE, therefore the actual value of the compound FF is 0.01 only (0.01 * 1), so choosing this index may not be the most efficient access path. However, if DB2 can collect the cardinality (distinct values) or most frequent concatenated values of CITY and STATE columns in the Address table, the optimizer can then calculate the compound filter factor accurately and choose a better access path.

Unless the proper statistics are collected, DB2 calculates the filter factors as if the columns in the Predicate are independent and are not correlated.

COLUMN CORRELATION ON MATCHING INDEX COLUMNS

Consider a table, TORDER, with 16 rows (see below), where the business rule says that each supplier can supply two parts only. This

table has two unique indexes defined on it, IORDER1 (ORDER_DEPT,ORDER_DT) and IORDER2 (SUPPLIER_ID,PART_ID, SUPPLY_DT):

```
ORDER_NUM  ORDER_DEPT  ORDER_DT    SUPPLIER_ID  PART_ID   SUPPLY_DT SUPPLY_QTY

164532     D001        1997-11-01  S00001       P001      1998-01-01    10
134652     D004        1998-01-15  S00003       P004      1998-02-10    20
700100     D013        1998-02-20  S00002       P004      1998-03-17     5
601321     D001        1998-01-23  S00001       P003      1998-02-16     7
167849     D006        1998-01-31  S00004       P002      1998-03-01     8
321016     D008        1997-12-15  S00002       P002      1998-01-21    20
109900     D010        1998-02-20  S00003       P003      1998-03-03     6
702311     D004        1998-03-01  S00004       P001      1998-04-01    10
463522     D020        1998-02-28  S00003       P003      1998-03-30    18
101034     D016        1998-02-23  S00002       P004      1998-03-10    20
006123     D004        1997-12-10  S00001       P001      1998-01-03     6
782123     D015        1998-03-01  S00004       P002      1998-03-13     5
432096     D010        1998-02-25  S00004       P002      1998-03-20     3
106010     D011        1997-12-31  S00001       P001      1998-01-07     9
234001     D003        1997-12-20  S00001       P001      1998-01-10     8
100600     D013        1998-01-01  S00001       P001      1998-01-12     7
```

The unique indexes on TORDER are:

```
IORDER1  (ORDER_DEPT, ORDER_DT)
IORDER2  (SUPPLIER_ID, PART_ID, SUPPLY_DT)
```

For a query, QUERY 5, the two compound predicates matching indexes IORDER1 and IORDER2 are (ORDER_DEPT= 'D001') and (SUPPLIER_ID='S00001' AND PART_ID='P001') respectively.

Query 5:

```
SELECT  *
FROM  TORDER
WHERE
  (SUPPLIER_ID = 'S00001'  AND  PART_ID = 'P001') | matching predicate 1

 AND
  (ORDER_DEPT  =  'D001')                         | matching predicate 2
```

Figure 1 illustrates DB2 estimates for compound filter factors and the estimated number of qualified leaf pages based on the assumption that columns SUPPLIER_ID and PART_ID are independent and therefore any supplier can supply any part, choosing index IORDER2 (with MATCHCOLS=2) over IORDER1 as a better access path. But we see that, based on actual filter factors and actual numbers of qualified leaf

	Index IORDER1	Index IORDER2
Matching predicates	ORDER_DEPT = 'D001'	SUPPLIER_ID = 'S00001' AND PART_ID = 'P001'
Number of matching columns	1	2
DB2 estimate for FFs for individual matching predicates	column ORDER_DEPT FIRSTKEYCARD=11 FF=1/11 = 0.0909	column SUPPLIER_ID COLCARD = 4 FF=1/4 = 0.25 column PART_ID COLCARD = 4 FF=1/4 = 0.25
DB2 estimate for compound FF	0.0909	0.25 * 0.25 = 0.0625
DB2 estimate for qualified leaf pages	0.0909 * 100 = 9.09	0.0625 * 100 = 6.25 DB2 chooses index IORDER2 as 6.25 < 9.09
Actual FF based on data distribution (number of rows satisfying predicates /total number of rows)	2/16 = 0.125	5/16 = 0.3125
Actual number of Qualified leaf pages	0.125 * 100 = 12.5 Index IORDER1 is better choice as 12.5 < 31.25	0.3125 * 100 = 31.25

Figure 1: Influence on access path by correlated columns

pages, index IORDER1 is a better choice even though MATCHCOLS for this predicate is only one.

COLUMN CORRELATION ON INDEX SCREENING COLUMNS

In index screening (ACCESSTYPE='I' and MATCHCOLS=0),

predicates are specified on index key columns, but they are not part of the matching columns. These predicates help in reducing the number of qualified data rows by scanning through the index. But, if the key columns used in the predicates are correlated, the index may filter more rows than the number estimated by the optimizer in DB2 V4. Consider Query 5 with indexes IORDER3 (ORDER_NUM, ORDER_DEPT) and IORDER4 (ORDER_NUM, SUPPLIER_ID, PART_ID). As we see, SUPPLIER_ID and PART_ID are correlated, so index IORDER4 may not filter as many rows as index IORDER3, although, based on estimated filter factors, the DB2 optimizer may choose index IORDER4.

COLUMN CORRELATION IN TABLE JOINS

If a table with correlated columns is joined with another table, the optimizer, based on the estimated value of the compound filter factor, may wrongly decide to choose a nested loop join method and may select this table as the outer table for a nested loop join.

SOLUTIONS FOR CORRELATIONS IN VERSION 4

If you know that DB2 is choosing an inefficient access path because of correlation, the only options available are to update the catalog statistics manually to influence access path or modify the SQL statement forcing the optimizer to discourage use of the inefficient index. As you know, neither of these alternatives is a good solution because the first one should be done with caution and statistics must be changed back to their original value after binding the program. The second one requires a program change and re-compilation.

GATHERING CORRELATION STATISTICS WITH RUNSTATS IN V5

DB2 V5 supports features to collect statistics about correlated columns and store them in catalog tables. Figure 2 shows columns in the SYSIBM.SYSCOLDIST (also in SYSCOLDISTSTATS) table, where DB2 stores statistics on concatenated key columns of an index. The RUNSTATS utility in Version 5 has new options to collect these statistics. Using these statistics, the optimizer gets accurate information

Column name	*Data Type*	*Description*
NAME	VARCHAR (18)	Name of column. For NUMCOLUMNS greater than 1, this column identifies the first column name of the set of columns associated with statistics
TYPE	CHAR (1)	Type of statistics collected - C for cardinality F for frequent value
CARDF	FLOAT	Number of distinct values for the column group. Valid for TYPE='C'
COLGROUPCOLNO	VARCHAR (254)	This field is an array of SMALLINT column numbers with a dimension equal to value in NUMCOLUMNS. It identifies a set of columns associated with the statistics.
NUMCOLUMNS	SMALLINT	Number of columns associated with statistics
FREQUENCYF	FLOAT	When this number is multiplied by 100, it gives the percentage of rows in the table with value specified in COLVALUE.
COLVALUE	VARCHAR (254)	Contains the data of a frequently occurring value.

Figure 2: Columns used in SYSIBM.SYSCOLDIST (and SYSIBM.SYSCOLDSISTSTATS)

to calculate the filter factor and decide on an appropriate access path.

In DB2 V4, RUNSTATS will INDEX records' cardinality values in FIRSTKEYCARD and FULLKEYCARD columns in SYSIBM.SYSINDEXES (and SYSIBM.SYSINDEXSTATS for partitioning indexes). It also collects the ten most frequent values for the first key column of the index in SYSIBM.SYSCOLDIST (and SYSIBM.SYSCOLDISTSTATS for partitioning index). It does not collect statistics on the concatenated key columns of a composite index. In DB2 V5, the RUNSTATS utility allows you to collect additional statistics on concatenated key columns of an index starting

from the first column and going up to the last but one key column. This means that, if the number of keys in an index is four, then additional statistics will be collected for concatenated values of the first and second key and concatenated values of the first, second, and third keys. These statistics can be gathered in terms of cardinality (number of distinct concatenated values) or most frequently concatenated values of key columns, depending on the RUNSTATS options KEYCARD and FREQVAL.

DB2 V5 RUNSTATS KEYCARD OPTION

With this option, RUNSTATS collects the cardinalities of each key column, concatenated with all the previous key columns. If, for example, there is an index with four key columns, then it will store:

- Cardinality of the second key column concatenated with the first key column in SYSIBM.SYSCOLDIST (and SYSCOLDISTSTATS).
- Cardinality of the third key column concatenated with the first and second key columns in SYSIBM.SYSCOLDIST (and SYSCOLDISTSTATS).

Cardinality of the first key column and first to fourth key columns are stored in FIRSTKEYCARDF and FULLKEYCARDF columns respectively in SYSIBM.SYSINDEXES (and SYSIBM. SYSINDEXPART) as was done in DB2 V4.

DB2 V5 RUNSTATS FREQVAL OPTION

With the FREQVAL option, RUNSTATS collects frequent value statistics. This option also includes two keywords – NUMCOLS, indicating the number of key columns (starting from the first key column) to be concatenated for collecting frequent value statistics, and COUNT, indicating the number of the most frequent value statistics to collect.

DB2 V5 RUNSTATS EXAMPLE

Consider table TORDER on which you want to collect statistics. The

RUNSTATS control statement is shown below:

```
RUNSTATS TABLESPACE   DB1.TS1
TABLE (TORDER)
INDEX (DB1.IORDER1 ,
       DB1.IORDER2    KEYCARD
                      FREQVAL  NUMCOLS  1  COUNT 16
                      FREQVAL  NUMCOLS  2  COUNT 16
                      FREQVAL  NUMCOLS  3  COUNT 16
      )
```

This statement will collect the following statistics:

- Cardinality on concatenation of the first and second columns of index IORDER2.
- Frequent values for:
  - The 16 most frequent values of SUPPLIER_ID, the first key column of index IORDER2.
  - The 16 most frequent values of SUPPLIER_ID concatenated with PART_ID, the first and second key columns of index IORDER2.
  - The 16 most frequent values of SUPPLIER_ID concatenated with PART_ID and SUPPLY_DT, the first three key columns of index IORDER2.
  - By default, the ten most frequent values of ORDER_DEPT, the first key of index IORDER1.

Besides these statistics, it will also collect FIRSTKEYCARDF and FULLKEYCARDF statistics for both the indexes (as done in DB2 V4).

In DB2 V5, we have additional columns in the catalog with the FLOAT data type. For example, CARDF, COLCARDF, etc. For our ease of reference, we can use an INTEGER function on these columns to extract values. Column COLGROUPCOLNO in SYSCOLDIST is a VARCHAR (254) and this column stores data as an array of SMALLINT columns containing column numbers. Column NUMCOLUMNS in SYSCOLDIST determines the occurrence of the SMALLINT values in the array.

The following illustrates a query to extract column name, type of statistics, cardinality, array of column numbers, frequency in percentage, and concatenated values:

```
SELECT TYPE,
        NAME AS COLUMN,
        STRIP(DIGITS(NUMCOLUMNS),LEADING,'Ø') AS NC,
        CASE
 WHEN CARDF ¬= -1 THEN
                STRIP(DIGITS(INTEGER(CARDF)),L,'Ø')

            ELSE
                '- NA -'
         END  AS  CARD,
         CASE
             WHEN NUMCOLUMNS = 2   THEN
                 HEX(SUBSTR(COLGROUPCOLNO,1,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,3,2))
             WHEN NUMCOLUMNS = 3   THEN
                 HEX(SUBSTR(COLGROUPCOLNO,1,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,3,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,5,2))
             WHEN NUMCOLUMNS = 4   THEN
                 HEX(SUBSTR(COLGROUPCOLNO,1,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,3,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,5,2))
                             ||' , '||
                 HEX(SUBSTR(COLGROUPCOLNO,7,2))
             ELSE
                 '- NA -'
         END  AS  GRPCOLNO,
         CASE
             WHEN FREQUENCYF ¬= -1       THEN
                 STRIP(DIGITS(INTEGER((FREQUENCYF * 1ØØ ))),L,'Ø')
             ELSE
                 '- NA -'
         END  AS  FRQ,
         CASE
             WHEN  ( NAME  = 'ORDER_DEPT' )
               AND ( FREQUENCYF ¬= -1  )    THEN
                 CASE
                     WHEN NUMCOLUMNS  =  1       THEN
                       SUBSTR(COLVALUE,1,4)
                     ELSE
```

```
                '- NA -'
            END
          WHEN  ( NAME  = 'SUPPLIER_ID' )
            AND ( FREQUENCYF ¬= -1  )    THEN
            CASE
                WHEN NUMCOLUMNS  =  1       THEN
                  SUBSTR(COLVALUE,1,6)
                WHEN NUMCOLUMNS  =  2       THEN
                  SUBSTR(COLVALUE,1,6)
                               ||' , '||
                  SUBSTR(COLVALUE,7,4)
                WHEN NUMCOLUMNS  =  3       THEN
                  SUBSTR(COLVALUE,1,6)
                               ||' , '||
                  SUBSTR(COLVALUE,7,4)
                               ||' , '||
                  HEX(SUBSTR(COLVALUE,11,2))            ||'/'||
                  HEX(SUBSTR(COLVALUE,13,1))            ||'/'||
                  HEX(SUBSTR(COLVALUE,14,1))
                ELSE
                  '- NA -'


            END
          ELSE
              '- NA -'
      END  AS  COLUMNS_VALUE
FROM  SYSIBM.SYSCOLDIST
WHERE TBOWNER = 'DB1'
AND   TBNAME  = 'TORDER'
ORDER  BY  1,2,3,4,5 DESC
  ;
```

The results of this query are shown in Figure 3. In these results, we can observe the following.

The first row has TYPE='C', indicating cardinality for the first key column SUPPLIER_ID (COLUMN=SUPPLIER_ID) concatenated with the second key column PART_ID (column numbers for SUPPLIER_ID and PART_ID are 4 and 5 respectively as shown in GRPCOLNO). The cardinality of concatenated key columns (SUPPLIER_ID and PART_ID) is 8. The next 10 rows correspond to the 10 most frequent values of the first key ORDER_DEPT, of index IORDER1. The next four rows correspond to the four most frequent values of the first key SUPPLIER_ID, of index IORDER2. Thereafter, the next eight rows correspond to the eight most frequent values of the second key PART_ID concatenated with first key of SUPPLIER_ID,

TYPE	COLUMN	NC	CARD	GRPCOLNO	FRQ	COLUMNS_VALUE
C	SUPPLIER_ID	2	8	0004 , 0005	- NA -	- NA -
F	ORDER_DEPT	1	- NA -	- NA -	18	D004
F	ORDER_DEPT	1	- NA -	- NA -	12	D001
F	ORDER_DEPT	1	- NA -	- NA -	12	D013
F	ORDER_DEPT	1	- NA -	- NA -	12	D010
F	ORDER_DEPT	1	- NA -	- NA -	6	D016
F	ORDER_DEPT	1	- NA -	- NA -	6	D015
F	ORDER_DEPT	1	- NA -	- NA -	6	D011
F	ORDER_DEPT	1	- NA -	- NA -	6	D008
F	ORDER_DEPT	1	- NA -	- NA -	6	D006
F	ORDER_DEPT	1	- NA -	- NA -	6	D003
F	SUPPLIER_ID	1	- NA -	- NA -	37	S00001
F	SUPPLIER_ID	1	- NA -	- NA -	25	S00004
F	SUPPLIER_ID	1	- NA -	- NA -	18	S00003
F	SUPPLIER_ID	1	- NA -	- NA -	18	S00002
F	SUPPLIER_ID	2	- NA -	0004 , 0005	31	S00001 , P001
F	SUPPLIER_ID	2	- NA -	0004 , 0005	18	S00004 , P002
F	SUPPLIER_ID	2	- NA -	0004 , 0005	12	S00002 , P004
F	SUPPLIER_ID	2	- NA -	0004 , 0005	12	S00003 , P003
F	SUPPLIER_ID	2	- NA -	0004 , 0005	6	S00001 , P003
F	SUPPLIER_ID	2	- NA -	0004 , 0005	6	S00004 , P001
F	SUPPLIER_ID	2	- NA -	0004 , 0005	6	S00003 , P004
F	SUPPLIER_ID	2	- NA -	0004 , 0005	6	S00002 , P002
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00004 , P002 , 1998/03/13
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00004 , P002 , 1998/03/01
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P001 , 1998/01/01
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P001 , 1998/01/12
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P001 , 1998/01/10
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P001 , 1998/01/07
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P001 , 1998/01/03
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00004 , P001 , 1998/04/01
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00003 , P004 , 1998/02/10
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00003 , P003 , 1998/03/30
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00003 , P003 , 1998/03/03
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00002 , P004 , 1998/03/17
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00002 , P004 , 1998/03/10
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00002 , P002 , 1998/01/21
F	SUPPLIER_ID	3	- NA -	0004 , 0005 , 0006	6	S00001 , P003 , 1998/02/16

Cardinality statistics in SYSIBM.SYSINDEXES

Index name	*FIRSTKEYCARDF*	*FULLKEYCARDF*
IORDER1	11	16
IORDER2	4	16

Figure 3: Correlation statistics in SYSIBM.SYSCOLDIST

of index IORDER2. The last 15 rows correspond to the 15 most frequent values of the third key SUPPLY_DT, concatenated with the second key PART_ID and the first key SUPPLIER_ID, of index IORDER2.

The frequent values percentage for SUPPLIER_ID concatenated with PART_ID is a maximum (31%) for the SUPPLIER_ID='S00001' and PART_ID='P001' combination.

USAGE OF KEYCARD AND FREQVAL OPTIONS IN RUNSTATS

Running RUNSTATS with these options is a CPU-intensive process and, as you increase the value of COUNT for collecting more and more frequent values (to get better filter factor estimates), the space requirement to collect these rows in the catalog also increases. Therefore you must choose these options carefully while running RUNSTATS.

If you are certain that the index has multicolumn correlated keys and your application has queries with partial matching (>1 and < total number of keys in index) MATCHCOLS equal predicates, you should consider using these options while running RUNSTATS on your indexes. Queries with only the first key matching equal predicates use FIRSTKEYCARDF in SYSINDEXES and FREQUENCYF in SYSCOLDIST tables to estimate filter factors. Similarly, queries with full key matching equal predicates use FULLKEYCARDF in SYSINDEXES and FREQUENCYF in SYSCOLDIST tables to estimate the filter factor.

If data is not uniformly distributed in key columns and the query has literals in the predicates or the query has host variables and it is re-optimized at run time (DB2 V5 feature), use the FREQVAL option, otherwise use the KEYCARD option. You may also consider a combination of both the options if required. You may consider using NUMCOLS for only those concatenated key columns which have been established to have correlation. The value of COUNT depends on the distribution of key data and the nature of queries. Before deciding the value of COUNT, consider getting the distribution statistics of correlated keys and then estimate a value of COUNT, which covers most of the non-uniformly distributed key values.

If the optimizer detects a match in the COLVALUE column for the concatenated value in SYSIBM.SYSCOLDIST, it establishes a filter factor from the FREQUENCYF value. If the optimizer does not find any match, then it assumes that the remaining concatenated values for the column group are uniformly distributed and it uses the value in column CARDF to find out the number of distinct values of the concatenated columns.

Consider Query 5 predicates for estimating filter factors. For compound predicate (SUPPLIER_ID = 'S00001' AND PART_ID = 'P001'), table SYSIBM.SYSCOLDIST has a frequent value (TYPE='F') row with COLVALUE = 'S00001P001' (COLUMN_VALUE='S00001, P001') having FREQUENCYF= 31 (FRQ = 31 %) for concatenated columns SUPPLIER_ID and PART_ID (Figure 3). Therefore, the DB2 V5 optimizer will estimate an FF of 0.31 for this compound predicate.

Similarly, for predicate (ORDER_DEPT = 'D001'), there exists a frequent value row in SYSCOLDIST with NAME='ORDER_DEPT', NUMCOLUMNS=1, and COLVALUE = 'D001' having FREQUENCYF = 12, and therefore, the DB2 V5 optimizer will estimate an FF of 0.12 for this single predicate.

As we can see, 0.12 is less than 0.31, so the optimizer will rightly choose index IORDER1 (with MATCHCOLS=1) over index IORDER2 (with MATCHCOLS = 2).

In another example, consider a predicate (ORDER_DEPT = 'D002'), which has no matching row in SYSIBM.SYSCOLDIST table with COLUMN = 'ORDER_DEPT', NUMCOLS = 1 and TYPE = 'F'. Also, the cardinality of column ORDER_DEPT is 11 (the value in FIRSTKEYCARDF of table SYSIBM.SYSINDEXES). As the ten most frequent values of ORDER_DEPT are available in SYSIBM.SYSCOLDIST and they do not include value 'D002', the remaining one distinct value (11-10) of column ORDER_DEPT is assumed to be uniformly distributed for the remaining 10% (100 - 90)18+12+12+12+6+6+6+6+6+6)) rows. Therefore, for predicate (ORDER_DEPT = 'D002'), the optimizer will calculate the FF as 1/1 * 0.1= 0.1.

DEGREE OF CORRELATION

In index IORDER2, the cardinality on concatenated values for SUPPLIER_ID and PART_ID is 8 and, if we compare this value with the FIRSTKEYCARDF value of 4 and FULLKEYCARDF value of 16 for index IORDER2, we can see that CARDF for SUPPLIER_ID concatenated with PART_ID is closer to FIRSTKEYCARDF as compared with FULLKEYCARDF. This indicates that key columns SUPPLIER_ID and PART_ID have a higher degree of correlation between them than key columns PART_ID and SUPPLY_DT.

RUNSTATS CONSTRAINT

In DB2 V5, RUNSTATS collects correlation statistics only on a concatenation of index key columns which include the first key column. If you wish to collect correlated statistics on key columns which do not include a first key column (for example key columns used in Index screening), you need to insert these rows manually in the SYSIBM.SYSCOLDIST table. Please keep in mind that any manual update to a catalog table requires extra caution and should be done temporarily and only for those queries needing these statistics.

CONCLUSION

Key column correlation statistics are needed for those queries with partial matching predicates on key columns that are correlated, and help the optimizer in deciding the most efficient access path for those queries. Also, there is always a CPU cost and extra space required for catalog table growth for collecting and storing these statistics through RUNSTATS.

Therefore, if a query has performance problem and it is due to correlated key columns, you may like to consider collecting concatenated key column statistics using RUNSTATS.

Sharad K Pande
Senior DBA
PriceWaterhouseCooper (USA)

Viewing DB2 dataset information – update

I just came across a condition which was not handled appropriately in the LISTCAT utility published in Issue 90, the April 2000 issue of *DB2 Update*, authored by me. The name of the article is *Viewing DB2 dataset information for a database using the LISTCAT command*.

When a dataset spans more than two volumes, the existing code will fail. I have corrected it and it is shown below:

```
/* rexx */
/*trace r     on if errors */
/*trace i     on always    */
/*trace o    off always    */
/* updated to take care of multiple extents on same    */
/* volume.              */
trace o
clear
gralc = 0
gruse = 0
usecyl = 0
alccyl = 0
rusecyl = 0
ralccyl = 0
PREFX = SYSVAR(SYSPREF)
/*   The 4 char VCAT HLQ and the subsystem ID together must form   */
/*   the full first level node of the DB2 datasets.               */
say
say 'Please input the 4 char VCAT HLQ  ...'
parse upper pull I_HLQ
say 'Enter the 4 char subsystem ID ...: '
parse upper pull I_sid
say 'Enter a Database name if you want to limit to one '
say ' Or Press Enter for all Databases .....'
parse upper pull I_dbname
I_HLQ = strip(I_HLQ)
I_sid = strip(I_sid)
I_dbname = strip(I_dbname)
P_CATHLQ = I_HLQ||I_sid
cd = date(U)
us_date = substr(cd,7,2)||substr(cd,1,2)||substr(cd,4,2)
ods_name = PREFX||"."||userid()||".OUTPUT."||P_CATHLQ
dbnode = substr(I_dbname,1,6)
if I_dbname = '' then
   nop
else
```

```
   ods_name = PREFX||"."||userid()||".OUTPUT."||P_CATHLQ||"."||dbnode
ods_name = ods_name||".D"||us_date
smry_ds = PREFX||'.'||userid()||'.SMRY.'||P_CATHLQ
if I_dbname = '' then
   nop
else
   smry_ds = PREFX||"."||userid()||".SMRY."||P_CATHLQ||"."||dbnode
smry_ds = smry_ds||".D"||us_date
call GETDSN
call GETDEF
xx=outtrap("zap.","*")
address tso "delete '"ods_name"'"
address tso "delete '"smry_ds"'"
xx=outtrap("OFF")
address tso "alloc f(opds) new unit(hsm) space(1,2)",
            "cyl reuse dsname('"ods_name"')",
            "dsorg(ps) blksize(13300) lrecl(133) recfm(f b)"
address tso "alloc f(smds) new unit(hsm) space(1,2)",
          "cyl release dsname('"smry_ds"')",
          "dsorg(ps) blksize(13300) lrecl(133) recfm(f b)"
If I_dbname = '' then
   P_CATNAM = P_CATHLQ||'.DSNDBD'
else
   P_CATNAM = P_CATHLQ||'.DSNDBD.'||I_dbname
P_CATNAM = strip(P_CATNAM)
x = outtrap("lsout.","*")
"Listcat level('"P_CATNAM"')"
x = outtrap("OFF")
i2=0
do  j=1 to lsout.0 by 2
  strng = strip(lsout.j)
  i2=i2+1
  parse VAR strng w1 w2 P_ddn.i2  w4.
end
fnd = 0
step1:
k=0
do z = 1 to i2
   if  (z//50) = 0 then
      say 'Processed 'z' members so far ...'
   DDN = strip(P_ddn.z)
   ADDRESS TSO
   parse var DDN a1 '.' a2 '.' O_dbname '.' O_obname '.' a3 '.' pno
   O_obname = strip(O_obname)
   O_dbname = strip(O_dbname)
   pno      = strip(pno)
   pno = substr(pno,2)
/* O_obname = substr(DDN,26,8)                                    */
/* O_dbname = substr(DDN,17,8)                                    */
   x = outtrap("lcout.","*")
```

```
  "Listcat entries('"DDN"') all"
  x = outtrap("OFF")
/*  get extents information from line 17  */
  strng = strip(lcout.17)
  parse VAR strng w1 w2 w3
  w3 = strip(w3)
  parse  VAR w3 dumy 8 exts
  N_exts = strip(exts,Leading,'-')
/*  get space type and HI-ARBA  from line 22  */
  strng = strip(lcout.22)
  parse VAR strng w1 w2
  w1=strip(w1)
  w2=strip(w2)
  parse var w1  dumy 11 spctyp
  parse var w2  dumy 10 hi_arba
  spctyp = strip(spctyp,Leading,'-')
  hi_arba = strip(hi_arba,Leading,'-')
/*  get PRIQTY and HI-URBA  from line 23  */
  strng = strip(lcout.23)
  parse VAR strng w1 w2
  w1=strip(w1)
  w2=strip(w2)
  parse var w1  dumy 10 priqty
  parse var w2  dumy 9 hi_urba
  priqty = strip(priqty,Leading,'-')
  hi_urba = strip(hi_urba,Leading,'-')
/*  get SECQTY              from line 24  */
  strng = strip(lcout.24)
  parse VAR strng w1
  w1=strip(w1)
  parse var w1 dumy 10 secqty
  secqty = strip(secqty,Leading,'-')
/*  get          RECSIZE    from line 26  */
  strng = strip(lcout.26)
  parse VAR strng w1 w2 w3
/*w1=strip(w1)                            */
  w2=strip(w2)
/*parse var w1 dumy 7 volser              */
  parse var w2 dumy 12 recsize
/*volser = strip(volser,Leading,'-')      */
  recsize = strip(recsize,Leading,'-')
/*  get tracks from line 30  */
/*strng = strip(lcout.30)                 */
/*parse VAR strng w1 w2 w3                */
/*w3=strip(w3)                            */
/*parse var w3 dumy 7 trcks               */
/*trcks = strip(trcks,Leading,'-')        */
/*  get vol names and tracks in each vol   */
  h = 0 ; t = 0
  do v = 26 to lcout.0
```

```
      strng = strip(lcout.v)
      parse VAR strng w1 rest
      if substr(strip(w1),1,6) = 'VOLSER' then
      do
         parse var w1 dumy 7 mvols
         mvols = strip(mvols,Leading,'-')
         h = h+1
         mulvols.h = mvols
         iterate
      end
      trkstr = strip(lcout.v)
      if pos('TRACKS--',trkstr) > 0 then
      do
         parse var trkstr w1 w2 w3
         w3=strip(w3)
         parse var w3 dumy 7 mtrks
         mtrks = strip(mtrks,Leading,'-')
         t = t + 1
         multrks.t = strip(mtrks)
         if t > h then
         do
            oh = h
            h = h +1
            mulvols.h = mulvols.oh
         end
      end
   end
/*say 'exts, spctyp, hi_arba, priqty' N_exts spctyp hi_arba priqty */
/*say 'secqty hi_urba volser recsize' secqty hi_urba volser recsize*/
   napgs = hi_arba / recsize
   nupgs = hi_urba / recsize
   spcuse = hi_urba/1024
   spcalc = hi_arba/1024
   cylalc = trunc(((spcuse*definc)+719)/720) * 720
   peruse = trunc((spcuse/spcalc*100),2)
   if cylalc <> 0 then
      newuse = trunc((spcuse/cylalc*100),2)
   else
      newuse = 99
/* peruse = substr(peruse,1,6)                  */
   if spctyp = 'CYLINDER' then
   do
      priqty = priqty*15
      secqty = secqty*15
      spctyp = 'CYL'
   end
   if spctyp = 'TRACKS' then
   do
      spctyp = 'TRK'
   end
```

```
priqty = trunc(priqty*49152/1024)
secqty = trunc(secqty*49152/1024)
  do while length(O_obname) < 8
     O_obname = ' '||O_obname
  end
  do while length(O_dbname) < 8
     O_dbname = ' '||O_dbname
  end
  do while length(priqty) < 10
     priqty = ' '||priqty
  end
  do while length(secqty) < 8
     secqty = ' '||secqty
  end
  do while length(N_exts) < 3
     N_exts = ' '||N_exts
  end
  do while length(nupgs) < 8
     nupgs = ' '||>upgs
  end
  do while length(spcuse) < 8
     spcuse = ' '||spcuse
  end
  do while length(spcalc) < 8
     spcalc = ' '||spcalc
  end
  do while length(cylalc) < 8
     cylalc = ' '||cylalc
  end
  do while length(peruse) < 6
     peruse = ' '||peruse
  end
  do while length(newuse) < 6
     newuse = ' '||>ewuse
  end
  if  h > 1 then
     volser = '******'
  if  h = 1 then
  do
      volser = mulvols.1
      volname = volser
      Call SUMVOLS
  end
  if substr(O_obname,7,1) = 'X' then
     ixmrkr = 'I'
  else
     ixmrkr = 'T'
  if pno > 1 then
  do
     oldstr = out.k
```

```
      out.k = overlay('P',oldstr,133)
      prtind = 'P'
   end
   else
     prtind = 'N'
   k = k+1
   out.k = O_dbname||' '||O_obname||'  '||pno||' '||volser
   out.k = out.k||' '||>upgs||' '||priqty||' '||secqty||'   '||N_exts
   out.k = out.k||' '||spcalc|| ' '||spcuse ||' 'peruse
   out.k = out.k||' '||cylalc||'  DEFAULT  '||>ewuse
   out.k = out.k||'  '||pno||'  'O_obname' '||ixmrkr||prtind
   if h>1 then
   do
     do j = 1 to t
        k=k+1
/*      say '***.multrks ' multrks.j O_obname  O_dbname        */
        multrks.j = trunc(multrks.j * 49152/1Ø24)
        volname = mulvols.j
        spcalc = multrks.j
        cylalc = Ø
        spcuse = Ø
        Call SUMVOLS
        do  while length(multrks.j) < 1Ø
            multrks.j = ' '||multrks.j
        end
        out.k = '                        '||mulvols.j
        out.k = out.k||'                                  '
        out.k = out.k||'  '||multrks.j
     end
   end
end
hdr.1 = '  DBNAME   OBJECT PART VOLSER    NUPGS       PQTY     SQTY'
hdr.1 = hdr.1||'  EXTS   SPCALC   SPCUSE   %USE    NPQTY    NSQTY'
hdr.1 = hdr.1||'   N%use PART    OBNAME'
hdr.2 = '-------------------------------------------------------'
hdr.2 = hdr.2||'-----------------------------------------------'
hdr.2 = hdr.2||'----------------------------'
"execio * diskw opds (stem hdr. "
"execio * diskw opds (FINIS  stem out. "
say 'Output written to 'ods_name
address tso "free ddname(opds)"
Call PRINT_SMRY
say 'Summary written to 'smry_ds
address tso "free ddname(smds)"
exit
SUMVOLS:
gruse = gruse + spcuse
gralc = gralc + spcalc
usecyl = usecyl + (spcuse/72Ø)
rusecyl = rusecyl + (trunc((spcuse+719)/72Ø))
alccyl = alccyl + (spcalc/72Ø)
```

```
ralccyl = ralccyl + (trunc((spcalc+719)/720))
if fnd = 0 then
do
   fnd=fnd + 1
   vollst.fnd = volname
   voltot.fnd = spcalc
   newtot.fnd = cylalc
   cyltot.fnd = spcuse/720
   return
end
else
do
   fndflg = 0
   do p = 1 to fnd
      if  vollst.p = volname then
      do
          voltot.p = voltot.p + spcalc
          newtot.p = newtot.p + cylalc
          cyltot.p = cyltot.p + (spcuse/720)
          fndflg = 1
      end
   end
   if  fndflg = 0 then
   do
      fnd=fnd+1
      vollst.fnd = volname
      voltot.fnd = spcalc
      newtot.fnd = cylalc
      cyltot.fnd = (spcuse/720)
      fndflg = 1
   end
end
return
PRINT_SMRY:
do g = 1 to fnd
   voltot.g = strip(voltot.g)
   newtot.g = strip(newtot.g)
   cyltot.g = strip(cyltot.g)
   voluse.g =  trunc(((newtot.g/voltot.g)*100),2)
   do while length(newtot.g) < 12
      newtot.g = ' '||>ewtot.g
   end
   do while length(voltot.g) < 12
      voltot.g = ' '||voltot.g
   end
   do while length(cyltot.g) < 12
      cyltot.g = ' '||cyltot.g
   end
/* smry.g = vollst.g||'     '||voltot.g||' '||>ewtot.g||'  'voluse.g */
   smry.g = vollst.g||'     '||voltot.g
   smry.g = smry.g||' '||cyltot.g
```

```
end
shdr.1 = 'VOLUME NAME   SPC ALLOC    SPC-U CYL'
shdr.2 = '-------------------------------------------------------'
"execio * diskw smds (stem shdr. "
ftr.1  = '-------------------------------------------------------'
ftr.2  = 'Total space used = 'gruse
ftr.3  = 'Total space allc = 'gralc
ftr.4  = 'Total cyls. used = 'usecyl
ftr.5  = 'Total cyls. allc = 'alccyl
ftr.6  = 'Total rounded cyls. used = 'rusecyl
ftr.7  = 'Total rounded cyls. allc = 'ralccyl
"execio * diskw smds (stem smry. "
"execio * diskw smds (stem ftr. FINIS"
return
GETDSN:
say
say 'Please enter output dataset name or Press Enter to ...'
say '   Use default dataset 'ods_name
say ' *** Note that the output dataset will be deleted if it exists ***'
pull I_dsname
upper I_dsname
I_dsname = strip(I_dsname,Both,"'")
I_dsname= strip(I_dsname)
if I_dsname = '' then
   nop
else
   ods_name = I_dsname
return
GETDEF:
definc = 30
Say  'Give the default percentage increase over the used quantity  '
Say  '  Or Press Enter for default (30) ...'
pull I_definc
upper I_definc
I_definc = strip(I_definc)
if strip(I_definc) = ''  then
   I_definc = definc
else
   definc = I_definc
if strip(I_definc) < 1 | strip(I_definc) > 100 then
do
   say ' ***  Error ***  Percentage must be between 1 and 100 '
   say
   SIGNAL GETDEF
end
definc = 1 + (definc/100)
return
```

Jaiwant K Jonathan
DB2 DBA (USA)

Simplifying occasional, regular, and periodic tasks of the DBA

One of main requirements at our installation is full availability of data to our customers on a 24x7 schedule. Under these conditions the 'maintenance window' is really narrow and is limited to the periods of the lowest system activity. The following procedure is designed to simplify some occasional, regular, and periodic tasks of the database administrator that are assumed to be done at the most suitable time.

Regular maintenance includes use of COPY, RUNSTATS, and STOSPACE utilities, as well as execution of REBIND of all packages and plans. Since at our installation image copies and archive logs are stored on tape and DFSMSrmm (Removable Media Manager) is the tape management system, we periodically execute jobs generated in order to clear outdated information from, and synchronize, the SYSIBM.SYSCOPY catalog table, bootstrap datasets, and RMM inventory of tape volumes used by DB2.

Our installation is currently using DB2 OS/390 Version 5 with production and test subsystems installed. The maintenance procedure is used in both environments and includes:

- MAINTR0 – application start-up REXX EXEC.
- MAINTR1 – regular maintenance REXX EXEC.
- MAINTR2 – periodic maintenance REXX EXEC.
- MAINTP1 – input panel.
- MAINTS1 – skeleton for full-image copy of catalog, directory, QMF, and some system tablespaces.
- MAINTS2 – skeleton for RMM extract.

Assumptions:

- Image copies and archive logs are stored on tape.
- DFSMSrmm is installed.

- Program SQLISPF, a modified version of SQLSPDB published in the January 1998 issue of *DB2 Update*, is available (code provided in this article).

Parameter descriptions follow (designed to our internal standards. Tailoring should be done only in panel MAINTP1).

Required:

- DB2 system – DB2 subsystem name (DSN or DBT).
- DB2 load – name of DB2 load modules library.
- Tape – tape unit type.
- Date – date before which deleting or scratching is done (given in the form *yyyymmdd* and used in options 6, 7, and 9).

Optional (used in options 1, 2, 6, and 9 to narrow selection. Any one can be specified. If not specified, relate to all tablespaces):

- Dbname – fully or partially specified database name.
- Tsname – fully or partially specified tablespace name.

Generated (relating to corresponding DB2 subsystem):

- BSDS01, BSDS02 – bootstrap dataset names.
- User data copy prefix – standard qualifier(s) for image copy datasets of user tablespaces.
- Catalog copy prefix – standard qualifier(s) for image copy datasets of catalog, directory, QMF, and some system tablespaces.
- Archive log prefix – partially specified (eg without log dataset number).

Note: option 3 generates a job for rebinding all packages and plans, while the STOSPACE utility (option 4) processes all storage groups for the chosen subsystem.

Option 7 is used for deleting all archive log datasets created before the specified date from the bootstrap datasets.

It should be emphasized that options 8 and 9 need to be executed in

sequence, because the job generated by option 8 forms dataset *&userid..RMM.PRIV*, which is used as input for the job generated by option 9.

Option 8 uses utility EDGHSKP and several REXX EXECs provided by DFSMSrmm to produce a report of all tape datasets and related tape volumes used at the installation. Based on that report and input parameters, option 9 deletes all information recorded by DFSMSrmm about volumes created before a specified date. If the tablespace or database name is fully or partially specified on the input panel, deletion is done only for volumes that contain image copies of the related tablespaces in that DB2 subsystem. Care should be taken not to delete a multi-file volume if it contains datasets that you want to keep. If neither database name nor tablespace name is given, volumes for all image copy datasets and archive log datasets for the chosen DB2 subsystem (before a specified date) are deleted from the DFSMSrmm inventory.

MAINTR0

```
/* rexx MAINTRØ */
address ispexec 'select panel(maintp1)'

MAINTP1

)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- MAINTENANCE -$-$-[
%Command ===><Z[                                        %Scroll ===><Z   [
+DB2 system ===><Z   [
+DB2 load   ===><Z                                  [
+TAPE       ===><Z     [
+DBNAME     ===><Z          [
+TSNAME     ===><Z          [
+DATE       ===><Z      [(yyyymmdd - date filter for modify,scratch,...)
+BSDSØ1     ===><Z                                  [
+BSDSØ2     ===><Z                                  [

+User data copy prefix ===>!Z                          [
+Catalog   copy prefix ===>!Z                          [
```

```
+Archive   log  prefix ===>!Z                          [

                 1. Generate JCL for full copy user's data
                 2. Generate JCL for runstats
                 3. Generate JCL for rebind
                 4. Generate JCL for stospace
                 5. Generate JCL for full copy catalog
                 6. Generate JCL for modify
                 7. Generate JCL for delete archive logs
                 8. Generate JCL for RMM extract
                 9. Generate JCL for RMM scratch
)INIT
   .ZVARS = '(ZCMD ZSCR DSN8SSID db2load mtape dbfilt tsfilt +
             mdat bsds1 bsds2 copyqu copyqc copyqa)'
  &ZSCR   = CSR
  .CURSOR = ZCMD
  VGET (DSN8SSID db2load mtape dbfilt tsfilt mdat bsds1 bsds2) SHARED
  VGET (copyqu copyqc copyqa) SHARED
  IF (&DSN8SSID = &Z)
    &DSN8SSID = DSN
  IF (&db2load = &Z)
    &db2load = DSN510.SDSNLOAD
  IF (&mtape = &Z)
    &mtape = TAPE
  IF (&mdat = &Z)
    &mdat = 20001231
  IF (&bsds1 = &Z)
    &bsds1 = DSNC510.BSDS01
  IF (&bsds2 = &Z)
    &bsds2 = DSNC510.BSDS02
  IF (&copyqu = &Z)
    &copyqu = BANKP.IMAGCOPY
  IF (&copyqc = &Z)
    &copyqc = DSN510.IMAGCOPY
  IF (&copyqa = &Z)
    &copyqa = DSNC510.ARCHLOG
)PROC
  IF (&DSN8SSID = DSN)
    &bsds1 = DSNC510.BSDS01
    &bsds2 = DSNC510.BSDS02
    &copyqu = BANKP.IMAGCOPY
    &copyqc = DSN510.IMAGCOPY
    &copyqa = DSNC510.ARCHLOG
  IF (&DSN8SSID = DBT)
    &bsds1 = DBTC510.BSDS01
    &bsds2 = DBTC510.BSDS02
    &copyqu = BANKT.IMAGCOPY
    &copyqc = DBT510.IMAGCOPY
    &copyqa = DBTC510.ARCHLOG
  VER (&ZCMD,RANGE,1,9)
```

```
  VER (&DSN8SSID,NONBLANK,LIST,DSN,DBT)
  VER (&mtape,NONBLANK)
  VER (&db2load,NONBLANK)
  VER (&mdat,NONBLANK)
  VER (&bsds1,NONBLANK)
  VER (&copyqu,NONBLANK)
  VER (&copyqc,NONBLANK)
  VER (&copyqa,NONBLANK)
  &S = &ZCMD
  VPUT (DSN8SSID db2load mtape dbfilt tsfilt S) SHARED
  VPUT (mdat bsds1 bsds2) SHARED
  VPUT (copyqu copyqc copyqa) SHARED
  &ZSEL = TRANS(TRUNC(&ZCMD,'.')
              1,'CMD(MAINTR1)'
              2,'CMD(MAINTR1)'
              3,'CMD(MAINTR1)'
              4,'CMD(MAINTR1)'
              5,'CMD(MAINTR1)'
              6,'CMD(MAINTR2)'
              7,'CMD(MAINTR2)'
              8,'CMD(MAINTR2)'
              9,'CMD(MAINTR2)'
              '  ',' '
              *,'?')
)END
```

MAINTR1

```
/* REXX - MAINTR1 *****************************************************/
/* TITLE    :  REGULAR MAINTENANCE                                    */
/* TRACE I                                                            */
  signal on error
/* ****************************************************************** */
/* init values                                                        */
/* ****************************************************************** */
  ADDRESS ISPEXEC "VGET (DSN8SSID db2load dbfilt tsfilt mtape S) SHARED"
  ADDRESS ISPEXEC "VGET (copyqu copyqc copyqa) SHARED"
  userid   = userid()
  tick     = ''''
  outdsn   = tick||userid||'.MAINT1.CNTL'||tick
  DB2V = DSN8SSID
/* ****************************************************************** */
  if sysdsn(outdsn) = 'OK' then
    "alloc fi(ispfile) da("outdsn") shr "
  else do
    "alloc fi(ispfile) da("outdsn") new ",
    "  dsorg(ps) space(1,1) tracks",
    "  recfm(F B) lrecl(132) blksize(27984)"
  end
select
```

```
when S = 1 then do
 queue '//'||userid||'1 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
 queue '//      TIME=144Ø,REGION=4M'
 queue '//*********************************************'
 queue '//* Full Image Copy User Data                  '
 queue '//*********************************************'
 queue '//JOBLIB  DD  DISP=SHR,DSN='||db2load
 queue "//STEPØØØ1 EXEC PGM=DSNUTILB,PARM='"||DSN8SSID||",ICOPYUD'"
 queue '//SYSPRINT DD SYSOUT=*                          '
 queue '//SYSUDUMP DD SYSOUT=*                          '
 "execio 9 diskw ispfile"
   SQLQUERY = "SELECT DBNAME, ",
                    "NAME ",
              "FROM SYSIBM.SYSTABLESPACE ",
              "WHERE DBNAME NOT LIKE 'DSN%' AND ",
                    "DBNAME NOT LIKE 'DSQ%' AND ",
                    "DBNAME NOT IN ('ADBDC', 'CELDIAL', 'RAADB', ",
                                  "'RDBIDB1', 'RDBIDB2', 'RDBIDB3', ",
                                  "'DBEDB1', 'DBEDB2') ",
                "AND DBNAME LIKE '"||dbfilt||"%' AND ",
                    "NAME LIKE '"||tsfilt||"%' ",
              "ORDER BY 1, 2";
   ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
   if  rc <> Ø then say 'Error'
   if  _nrows = Ø then say 'No record found'
   do i = 1 to _nrows
       dbname = strip(substr(DBNAME.i, 1, 8))
       tsname = strip(substr(NAME.i, 1, 8))
       copyname = strip(copyqu)||'.'||dbname||'.'||tsname
       if i = 1 then do
         queue '//SYSCO'||i||' DD UNIT='||mtape||',DISP=(NEW,KEEP),'
         queue '//            DSN='||copyname||',VOL=(,,,5Ø),'
         queue '//            LABEL=('||i||',SL)'
       end
       else do
         queue '//SYSCO'||i||' DD UNIT=AFF=SYSCO'||i - 1||,
               ',VOL=(,RETAIN,REF=*.SYSCO'||i - 1||'),'
         queue '//            DSN='||copyname||','
         queue '//            LABEL=('||i||',SL),DISP=(NEW,KEEP)'
       end
      "execio 3 diskw ispfile"
   end
   queue '//SYSIN    DD  *'
  "execio 1 diskw ispfile"
   do i = 1 to _nrows
      dbname = strip(DBNAME.i)
      tsname = strip(NAME.i)
      queue '   COPY TABLESPACE '||dbname||'.'||tsname||,
            ' COPYDDN SYSCO'||i
     "execio 1 diskw ispfile"
   end
```

```
end
when S = 2 then do
  queue '//'||userid||'2 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
  queue '//      TIME=144Ø,REGION=4M'
  queue '//*********************************************'
  queue '//* Run Statistics                             '
  queue '//*********************************************'
  queue "//STEPØØØ1 EXEC DSNUPROC,SYSTEM="||DSN8SSID||",UID='STAT',"||,
        "UTPROC=''"
  queue '//DSNUPROC.SYSIN    DD *'
  "execio 7 diskw ispfile"
  SQLQUERY = "SELECT DBNAME, ",
                   "NAME ",
             "FROM SYSIBM.SYSTABLESPACE ",
             "WHERE DBNAME NOT LIKE 'DSN%' AND ",
                   "DBNAME NOT LIKE 'DSQ%' AND ",
                   "DBNAME NOT IN ('ADBDC', 'CELDIAL', 'RAADB', ",
                                  "'RDBIDB1', 'RDBIDB2', 'RDBIDB3', ",
                                  "'DBEDB1', 'DBEDB2') ",
               "AND DBNAME LIKE '"||dbfilt||"%' AND ",
                   "NAME LIKE '"||tsfilt||"%' ",
             "ORDER BY 1, 2";
  ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
  if  rc <> Ø then say 'Error'
  if  _nrows = Ø then say 'No record found'
  do i = 1 to _nrows
    dbname = strip(substr(DBNAME.i, 1, 8))
    tsname = strip(substr(NAME.i, 1, 8))
    queue 'RUNSTATS TABLESPACE '||dbname||'.'||tsname||' ',
          'TABLE(ALL) INDEX(ALL)'
   "execio 1 diskw ispfile"
  end
 end
when S = 3 then do
  queue '//'||userid||'3 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
  queue '//      TIME=144Ø,REGION=4M'
  queue '//*********************************************'
  queue '//* Rebind packages and plans                  '
  queue '//*********************************************'
  queue '//STEPØØØ1 EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø'
  queue '//SYSTSPRT DD SYSOUT=*'
  queue '//SYSPRINT DD SYSOUT=*'
  queue '//SYSUDUMP DD SYSOUT=*'
  queue '//SYSIN    DD DUMMY'
  queue '//SYSTSIN  DD *'
  queue ' DSN SYSTEM('||DSN8SSID||')'
  queue '   REBIND PACKAGE(*)'
  queue '   REBIND PLAN(*)'
  queue ' END'
  "execio 15 diskw ispfile"
end
```

```
 when S = 4 then do
   queue '//'||userid||'4 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
   queue '//      TIME=144Ø,REGION=4M'
   queue '//*********************************************'
   queue '//* Storage Space                              '
   queue '//*********************************************'
   queue "//STEPØØØ1 EXEC DSNUPROC,SYSTEM="||DSN8SSID||",UID='STOSP'"||,
         ",UTPROC=''"
   queue '//SYSPRINT  DD SYSOUT=*'
   queue '//DSNUPROC.SYSIN    DD *'
   queue '  STOSPACE STOGROUP(*)'
   "execio 9 diskw ispfile"
 end
 when S = 5 then do
  ADDRESS ISPEXEC "FTOPEN"
  ADDRESS ISPEXEC "FTINCL MAINTS1"
  ADDRESS ISPEXEC "FTCLOSE"
 end
 otherwise
end
if S < 5 then do
  queue '//'
  "execio 1 diskw ispfile"
  "execio Ø diskw ispfile(finis"
end
signal off error
"free  fi(ispfile)"
"ispexec edit dataset("outdsn")"
signal on error
"ispexec lmerase dataset("outdsn")"
exit
error:
say 'error on line:' sigl '  ,rc:' rc
exit
```

MAINTR2

```
/* REXX - MAINTR2 ****************************************************/
/* TITLE    :  PERIODIC MAINTENANCE                                  */
/* TRACE I                                                           */
   signal off error
/* ***************************************************************** */
/* init values                                                       */
/* ***************************************************************** */
  ADDRESS ISPEXEC "VGET (DSN8SSID db2load dbfilt tsfilt S) SHARED"
  ADDRESS ISPEXEC "VGET (mdat, bsds1, bsds2) SHARED"
  ADDRESS ISPEXEC "VGET (copyqu, copyqc, copyqa) SHARED"
  NUM_OF_DAYS.1 = 31;
  NUM_OF_DAYS.3 = 31;
  NUM_OF_DAYS.4 = 3Ø;
```

```
  NUM_OF_DAYS.5 = 31;
  NUM_OF_DAYS.6 = 3Ø;
  NUM_OF_DAYS.7 = 31;
  NUM_OF_DAYS.8 = 31;
  NUM_OF_DAYS.9 = 3Ø;
  NUM_OF_DAYS.1Ø = 31;
  NUM_OF_DAYS.11 = 3Ø;
  NUM_OF_DAYS.12 = 31;
  yyyy = substr(mdat, 1, 4)
  mm   = substr(mdat, 5, 2)
  dd   = substr(mdat, 7, 2)
  if yyyy // 4ØØ <> Ø  &  yyyy // 4 = Ø  then /* leap year? */
    NUM_OF_DAYS.2 = 29
  else
    NUM_OF_DAYS.2 = 28;
  ddd = Ø;
  do i = 1 to mm - 1;
    ddd = ddd + NuM_OF_DAYS.I;
  end;
  ddd = ddd + dd;
  if ddd < 1ØØ then
    datyyyyddd = yyyy||'Ø'||ddd;
  else
    datyyyyddd = yyyy||ddd;
  userid   = userid()
  tick     = ''''
  outdsn   = tick||userid||'.MAINT2.CNTL'||tick
  DB2V = DSN8SSID
/* ******************************************************************* */
  if sysdsn(outdsn) = 'OK' then
    "alloc fi(ispfile) da("outdsn") shr "
  else do
    "alloc fi(ispfile) da("outdsn") new ",
    "  dsorg(ps) space(1,1) tracks",
    "  recfm(F B) lrecl(132) blksize(27984)"
  end
select
 when S = 6 then do
  queue '//'||userid||'6 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
  queue '//      TIME=144Ø,REGION=4M'
  queue '//*********************************************'
  queue '//* Modify SYSCOPY                             '
  queue '//*********************************************'
  queue "//STEPØØØ1 EXEC DSNUPROC,SYSTEM="||DSN8SSID||",UID='MODIF',"||,
        "UTPROC=''"
  queue '//DSNUPROC.SYSIN    DD *'
  "execio 7 diskw ispfile"
  SQLQUERY = "SELECT DBNAME, ",
                     "NAME ",
             "FROM SYSIBM.SYSTABLESPACE ",
             "WHERE DBNAME LIKE '"||dbfilt||"%' AND ",
```

```
                "NAME LIKE '"||tsfilt||"%' ",
           "ORDER BY 1, 2";
 ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
 if  rc <> Ø then say 'Error'
 if  _nrows = Ø then say 'No record found'
 do i = 1 to _nrows
   dbname = strip(DBNAME.i)
   tsname = strip(NAME.i)
   queue ' MODIFY RECOVERY TABLESPACE '||dbname||'.'||tsname||,
         ' DELETE DATE('||mdat||")"
  "execio 1 diskw ispfile"
 end
end
when S = 7 then do
 queue '//'||userid||'7 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
 queue '//      TIME=144Ø,REGION=4M'
 queue '//*********************************************'
 queue '//* Prerequisite:   STOP DB2                  '
 queue '//*********************************************'
 queue '//JOBLIB   DD DSN='||db2load||',DISP=SHR'
 queue '//STEPØØØ1 EXEC PGM=DSNJUØØ3'
 queue '//SYSUT1   DD  DISP=OLD,DSN='||bsds1
 queue '//SYSUT2   DD  DISP=OLD,DSN='||bsds2
 queue '//SYSPRINT DD  SYSOUT=*'
 queue '//SYSUDUMP DD SYSOUT=*'
 queue '//SYSIN    DD  *'
 "execio 12 diskw ispfile"
  printlib=tick||userid||'.MAINT.SYSPRINT'||tick
  sysut1=bsds1
  "free fi(sysut1)"
  "free fi(sysprint)"
  "alloc fi(sysut1) da('"sysut1"') shr"
  if sysdsn(printlib) = 'OK' then
    "alloc fi(sysprint) da("printlib") shr "
  else
    "alloc fi(sysprint) da("printlib") new ",
    "  dsorg(ps) space(4,2) tracks",
    "  lrecl(125) blksize(6144) recfm(v b a)"
  signal on error
  say 'Press ENTER to continue'
  "CALL '"||db2load||"(DSNJUØ4)'"
  signal off error
  indsn=tick||userid||'.MAINT.SYSPRINT'||tick
  "free  fi(indd)"
  "alloc fi(indd) da("indsn") shr "
  "delstack"
  do forever
    "execio 1 diskr indd"
    if rc = 2 then leave
    pull inrec
    if index(inrec, copyqa) > Ø then do
```

```
        yddd = substr(inrec, 47, 4) || substr(inrec, 52, 3)
        if yddd < datyyyyddd then do
           start = 68
           l = 25
           pomvar = 'DELETE DSNAME='||substr(inrec, start, l)
           n = substr(inrec, 83, 1)
           "execio 1 diskr indd"
           pull inrec
           if rc = 2 then leave
           start = 84
           l = 6
           pomvar = pomvar||',COPY'||n||'VOL='||substr(inrec, start, l)
           queue pomvar
           "execio 1 diskw ispfile"
        end
     end
   end
end
when S = 8 then do
   ADDRESS ISPEXEC "FTOPEN"
   ADDRESS ISPEXEC "FTINCL MAINTS2"
   ADDRESS ISPEXEC "FTCLOSE"
end
when S = 9 then do
 pomvol = ''
 queue '//'||userid||'9 JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid||','
 queue '//      TIME=1440,REGION=4M'
 queue '//*********************************************'
 queue '//* RMM SCRATCH - REALLY                      '
 queue '//*********************************************'
 queue '//STEP0001 EXEC PGM=IKJEFT01'
 queue '//SYSPRINT DD  SYSOUT=*'
 queue '//SYSTSPRT DD SYSOUT=*'
 queue '//SYSTSIN  DD  *'
 "execio 9 diskw ispfile"
  indsn=tick||userid||'.RMM.PRIV'||tick
  if dbfilt <> '' then do
    copyqu = strip(copyqu)||'.'||dbfilt
  end
  if tsfilt <> '' then do
    copyqu = strip(copyqu)||'.'||tsfilt
  end
  "free  fi(indd)"
  "alloc fi(indd) da("indsn") shr "
  "delstack"
  do forever
    "execio 1 diskr indd"
    if rc = 2 then leave
    pull inrec
    ind = 0
    if dbfilt <> '' | tsfilt <> '' then do
```

```
        if index(inrec, strip(copyqu)) > Ø then do
          ind = 1
        end
      end
      else do
        if index(inrec, strip(copyqu)) > Ø |,
           index(inrec, strip(copyqc)) > Ø |,
           index(inrec, strip(copyqa)) > Ø then do
          ind = 1
        end
      end
      if ind = 1 then do
         yyyy = substr(inrec, 121, 4)
         mm   = substr(inrec, 118, 2)
         dd   = substr(inrec, 115, 2)
         if yyyy // 4ØØ <> Ø  &  yyyy // 4 = Ø  then /* leap year? */
           NUM_OF_DAYS.2 = 29
         else
           NUM_OF_DAYS.2 = 28;
         ddd = Ø;
         do i = 1 to mm - 1;
           ddd = ddd + NuM_OF_DAYS.I;
         end;
         ddd = ddd + dd;
         if ddd < 1ØØ then
           yddd = yyyy||'Ø'||ddd;
         else
           yddd = yyyy||ddd;
         if yddd < datyyyyddd then do
           if pomvol <> substr(inrec, 3, 6) then do
              pomvol = substr(inrec, 3, 6)
              pomvar = '    RMM DV  '||substr(inrec, 3, 6)||,
                       ' EJECT(CONVENIENCE) RELEASE'
              queue pomvar
              "execio 1 diskw ispfile"
           end
         end
      end
   end
 end
 otherwise
end
if S <> 8 then do
  queue '//'
  "execio 1 diskw ispfile"
  "execio Ø diskw ispfile(finis"
end
if S = 7 | S = 9 then do
  "free fi(indd)"
  "ispexec lmerase dataset("indsn")"
end
```

```
"free  fi(ispfile)"
"ispexec edit dataset("outdsn")"
signal on error
"ispexec lmerase dataset("outdsn")"
exit
error:
say 'error on line:' sigl '  ,rc:' rc
exit
```

MAINTS1

```
)CM -------------------------------------------
)CM FULL IMAGE COPY CATALOG
)CM -------------------------------------------
//&userid.5 JOB MSGCLASS=X,TIME=144Ø,REGION=4M,NOTIFY=&userid
//JOBLIB   DD  DISP=SHR,DSN=&db2load
//*********************************************
//* FULL IMAGE COPY CATALOG
//*********************************************
//STEPØØØ1 EXEC PGM=DSNUTILB,PARM='&DSN8SSID,ICOPYCA1'
//SYSCOØØ1 DD UNIT=&mtape,DISP=(NEW,PASS),
//         DSN=&copyqc..RAADB.RAATS1,VOL=(,,,5Ø),
//         LABEL=(1,SL)
//SYSCOØØ2 DD UNIT=AFF=SYSCOØØ1,VOL=(,RETAIN,REF=*.SYSCOØØ1),
//         DSN=&copyqc..RAADB.RAATS2,
//         LABEL=(2,SL),DISP=(NEW,KEEP)
//SYSCOØØ3 DD UNIT=AFF=SYSCOØØ2,VOL=(,RETAIN,REF=*.SYSCOØØ2),
//         DSN=&copyqc..RAADB.RAATS3,
//         LABEL=(3,SL),DISP=(NEW,KEEP)
//SYSCOØØ4 DD UNIT=AFF=SYSCOØØ3,VOL=(,RETAIN,REF=*.SYSCOØØ3),
//         DSN=&copyqc..RAADB.RAATS4,
//         LABEL=(4,SL),DISP=(NEW,KEEP)
//SYSCOØØ5 DD UNIT=AFF=SYSCOØØ4,VOL=(,RETAIN,REF=*.SYSCOØØ4),
//         DSN=&copyqc..DSQDBCTL.DSQTSCT1,
//         LABEL=(5,SL),DISP=(NEW,KEEP)
//SYSCOØØ6 DD UNIT=AFF=SYSCOØØ5,VOL=(,RETAIN,REF=*.SYSCOØØ5),
//         DSN=&copyqc..DSQDBCTL.DSQTSCT2,
//         LABEL=(6,SL),DISP=(NEW,KEEP)
//SYSCOØØ7 DD UNIT=AFF=SYSCOØØ6,VOL=(,RETAIN,REF=*.SYSCOØØ6),
//         DSN=&copyqc..DSQDBCTL.DSQTSCT3,
//         LABEL=(7,SL),DISP=(NEW,KEEP)
//SYSCOØØ8 DD UNIT=AFF=SYSCOØØ7,VOL=(,RETAIN,REF=*.SYSCOØØ7),
//         DSN=&copyqc..DSQDBCTL.DSQTSGOV,
//         LABEL=(8,SL),DISP=(NEW,KEEP)
//SYSCOØØ9 DD UNIT=AFF=SYSCOØØ8,VOL=(,RETAIN,REF=*.SYSCOØØ8),
//         DSN=&copyqc..DSQDBCTL.DSQTSLOG,
//         LABEL=(9,SL),DISP=(NEW,KEEP)
//SYSCOØ1Ø DD UNIT=AFF=SYSCOØØ9,VOL=(,RETAIN,REF=*.SYSCOØØ9),
//         DSN=&copyqc..DSQDBCTL.DSQTSPRO,
//         LABEL=(1Ø,SL),DISP=(NEW,KEEP)
```

```
//SYSCØ011 DD UNIT=AFF=SYSCØ1Ø,VOL=(,RETAIN,REF=*.SYSCØ1Ø),
//         DSN=&copyqc..DSQDBCTL.DSQTSRDO,
//         LABEL=(11,SL),DISP=(NEW,KEEP)
//SYSCØ012 DD UNIT=AFF=SYSCØ11,VOL=(,RETAIN,REF=*.SYSCØ11),
//         DSN=&copyqc..DSQDBCTL.DSQTSSYN,
//         LABEL=(12,SL),DISP=(NEW,KEEP)
//SYSCØ013 DD UNIT=AFF=SYSCØ12,VOL=(,RETAIN,REF=*.SYSCØ12),
//         DSN=&copyqc..DSQDBDEF.DSQTSDEF,
//         LABEL=(13,SL),DISP=(NEW,KEEP)
//SYSCØ014 DD UNIT=AFF=SYSCØ13,VOL=(,RETAIN,REF=*.SYSCØ13),
//         DSN=&copyqc..DSQ1STBB.DSQ1STBT,
//         LABEL=(14,SL),DISP=(NEW,KEEP)
//SYSCØ015 DD UNIT=AFF=SYSCØ14,VOL=(,RETAIN,REF=*.SYSCØ14),
//         DSN=&copyqc..DSNRGFDB.DSNRGFTS,
//         LABEL=(15,SL),DISP=(NEW,KEEP)
//SYSCØ016 DD UNIT=AFF=SYSCØ15,VOL=(,RETAIN,REF=*.SYSCØ15),
//         DSN=&copyqc..DSNRLST.DSNRLSØ1,
//         LABEL=(16,SL),DISP=(NEW,KEEP)
//SYSCØ017 DD UNIT=AFF=SYSCØ16,VOL=(,RETAIN,REF=*.SYSCØ16),
//         DSN=&copyqc..DSNDBØ1.DBDØ1,
//         LABEL=(17,SL),DISP=(NEW,KEEP)
//SYSCØ018 DD UNIT=AFF=SYSCØ17,VOL=(,RETAIN,REF=*.SYSCØ17),
//         DSN=&copyqc..DSNDBØ1.SCTØ2,
//         LABEL=(18,SL),DISP=(NEW,KEEP)
//SYSCØ019 DD UNIT=AFF=SYSCØ18,VOL=(,RETAIN,REF=*.SYSCØ18),
//         DSN=&copyqc..DSNDBØ1.SPTØ1,
//         LABEL=(19,SL),DISP=(NEW,KEEP)
//SYSCØ02Ø DD UNIT=AFF=SYSCØ19,VOL=(,RETAIN,REF=*.SYSCØ19),
//         DSN=&copyqc..DSNDBØ6.SYSDBASE,
//         LABEL=(2Ø,SL),DISP=(NEW,KEEP)
//SYSCØ021 DD UNIT=AFF=SYSCØ2Ø,VOL=(,RETAIN,REF=*.SYSCØ2Ø),
//         DSN=&copyqc..DSNDBØ6.SYSDBAUT,
//         LABEL=(21,SL),DISP=(NEW,KEEP)
//SYSCØ022 DD UNIT=AFF=SYSCØ21,VOL=(,RETAIN,REF=*.SYSCØ21),
//         DSN=&copyqc..DSNDBØ6.SYSGPAUT,
//         LABEL=(22,SL),DISP=(NEW,KEEP)
//SYSCØ023 DD UNIT=AFF=SYSCØ22,VOL=(,RETAIN,REF=*.SYSCØ22),
//         DSN=&copyqc..DSNDBØ6.SYSGROUP,
//         LABEL=(23,SL),DISP=(NEW,KEEP)
//SYSCØ024 DD UNIT=AFF=SYSCØ23,VOL=(,RETAIN,REF=*.SYSCØ23),
//         DSN=&copyqc..DSNDBØ6.SYSPLAN,
//         LABEL=(24,SL),DISP=(NEW,KEEP)
//SYSCØ025 DD UNIT=AFF=SYSCØ24,VOL=(,RETAIN,REF=*.SYSCØ24),
//         DSN=&copyqc..DSNDBØ6.SYSPKAGE,
//         LABEL=(25,SL),DISP=(NEW,KEEP)
//SYSCØ026 DD UNIT=AFF=SYSCØ25,VOL=(,RETAIN,REF=*.SYSCØ25),
//         DSN=&copyqc..DSNDBØ6.SYSUSER,
//         LABEL=(26,SL),DISP=(NEW,KEEP)
//SYSCØ027 DD UNIT=AFF=SYSCØ26,VOL=(,RETAIN,REF=*.SYSCØ26),
//         DSN=&copyqc..DSNDBØ6.SYSSTR,
//         LABEL=(27,SL),DISP=(NEW,KEEP)
```

```
//SYSCØØ28 DD UNIT=AFF=SYSCØØ27,VOL=(,RETAIN,REF=*.SYSCØØ27),
//         DSN=&copyqc..DSNDBØ6.SYSVIEWS,
//         LABEL=(28,SL),DISP=(NEW,KEEP)
//SYSCØØ29 DD UNIT=AFF=SYSCØØ28,VOL=(,RETAIN,REF=*.SYSCØØ28),
//         DSN=&copyqc..DSNDBØ6.SYSSTATS,
//         LABEL=(29,SL),DISP=(NEW,KEEP)
//SYSCØØ3Ø DD UNIT=AFF=SYSCØØ29,VOL=(,RETAIN,REF=*.SYSCØØ29),
//         DSN=&copyqc..DSNDBØ6.SYSDDF,
//         LABEL=(3Ø,SL),DISP=(NEW,KEEP)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD  *
   COPY TABLESPACE RAADB.RAATS1      COPYDDN SYSCØØØ1
   COPY TABLESPACE RAADB.RAATS2      COPYDDN SYSCØØØ2
   COPY TABLESPACE RAADB.RAATS3      COPYDDN SYSCØØØ3
   COPY TABLESPACE RAADB.RAATS4      COPYDDN SYSCØØØ4
   COPY TABLESPACE DSQDBCTL.DSQTSCT1 COPYDDN SYSCØØØ5
   COPY TABLESPACE DSQDBCTL.DSQTSCT2 COPYDDN SYSCØØØ6
   COPY TABLESPACE DSQDBCTL.DSQTSCT3 COPYDDN SYSCØØØ7
   COPY TABLESPACE DSQDBCTL.DSQTSGOV COPYDDN SYSCØØØ8
   COPY TABLESPACE DSQDBCTL.DSQTSLOG COPYDDN SYSCØØØ9
   COPY TABLESPACE DSQDBCTL.DSQTSPRO COPYDDN SYSCØØ1Ø
   COPY TABLESPACE DSQDBCTL.DSQTSRDO COPYDDN SYSCØØ11
   COPY TABLESPACE DSQDBCTL.DSQTSSYN COPYDDN SYSCØØ12
   COPY TABLESPACE DSQDBDEF.DSQTSDEF COPYDDN SYSCØØ13
   COPY TABLESPACE DSQ1STBB.DSQ1STBT COPYDDN SYSCØØ14
   COPY TABLESPACE DSNRGFDB.DSNRGFTS COPYDDN SYSCØØ15
   COPY TABLESPACE DSNRLST.DSNRLSØ1  COPYDDN SYSCØØ16
   COPY TABLESPACE DSNDBØ1.DBDØ1     COPYDDN SYSCØØ17
   COPY TABLESPACE DSNDBØ1.SCTØ2     COPYDDN SYSCØØ18
   COPY TABLESPACE DSNDBØ1.SPTØ1     COPYDDN SYSCØØ19
   COPY TABLESPACE DSNDBØ6.SYSDBASE  COPYDDN SYSCØØ2Ø
   COPY TABLESPACE DSNDBØ6.SYSDBAUT  COPYDDN SYSCØØ21
   COPY TABLESPACE DSNDBØ6.SYSGPAUT  COPYDDN SYSCØØ22
   COPY TABLESPACE DSNDBØ6.SYSGROUP  COPYDDN SYSCØØ23
   COPY TABLESPACE DSNDBØ6.SYSPLAN   COPYDDN SYSCØØ24
   COPY TABLESPACE DSNDBØ6.SYSPKAGE  COPYDDN SYSCØØ25
   COPY TABLESPACE DSNDBØ6.SYSUSER   COPYDDN SYSCØØ26
   COPY TABLESPACE DSNDBØ6.SYSSTR    COPYDDN SYSCØØ27
   COPY TABLESPACE DSNDBØ6.SYSVIEWS  COPYDDN SYSCØØ28
   COPY TABLESPACE DSNDBØ6.SYSSTATS  COPYDDN SYSCØØ29
   COPY TABLESPACE DSNDBØ6.SYSDDF    COPYDDN SYSCØØ3Ø
//*
//STEPØØØ2 EXEC PGM=DSNUTILB,PARM='&DSN8SSID,ICOPYCA2',COND=(4,LT)
//SYSCØØ31 DD UNIT=&mtape,
//         DSN=&copyqc..DSNDBØ1.SYSUTILX,
//         LABEL=(31,SL),
//         VOL=(,RETAIN,REF=*.STEPØØØ1.SYSCØØ3Ø),
//         DISP=(NEW,PASS)
//SYSCØØØ1 DD UNIT=&mtape,DISP=(OLD,KEEP),
//         DSN=&copyqc..RAADB.RAATS1
```

```
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD  *
   COPY TABLESPACE DSNDB01.SYSUTILX  COPYDDN SYSCO031
//*
//STEP0003 EXEC PGM=DSNUTILB,PARM='&DSN8SSID,ICOPYCA3',COND=(4,LT)
//SYSCO032 DD UNIT=&mtape,
//         DSN=&copyqc..DSNDB01.SYSLGRNX,
//         LABEL=(32,SL),
//         VOL=(,RETAIN,REF=*.STEP0002.SYSCO031),
//         DISP=(NEW,PASS)
//SYSCO031 DD UNIT=&mtape,DISP=(OLD,KEEP),
//         DSN=&copyqc..DSNDB01.SYSUTILX
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD  *
   COPY TABLESPACE DSNDB01.SYSLGRNX  COPYDDN SYSCO032
//*
//STEP0004 EXEC PGM=DSNUTILB,PARM='&DSN8SSID,ICOPYCA4',COND=(4,LT)
//SYSCO033 DD UNIT=&mtape,
//         DSN=&copyqc..DSNDB06.SYSCOPY,
//         LABEL=(33,SL),
//         VOL=(,RETAIN,REF=*.STEP0003.SYSCO032),
//         DISP=(NEW,KEEP)
//SYSCO032 DD UNIT=&mtape,DISP=(OLD,KEEP),
//         DSN=&copyqc..DSNDB01.SYSLGRNX
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD  *
   COPY TABLESPACE DSNDB06.SYSCOPY   COPYDDN SYSCO033
//*
```

MAINTS2

```
)CM -------------------------------------------
)CM RMM SCRATCH
)CM   RMM.MESSAGES
)CM       SPACE=(TRK,(1,1)),
)CM       DCB=(RECFM=VB,LRECL=124,BLKSIZE=27998)
)CM   RMM.REPORT.EXTRACT
)CM       SPACE=(TRK,(300,1)),
)CM       DCB=(RECFM=VB,LRECL=719,BLKSIZE=27998)
)CM   RMM.EXTENDED.REPORT.EXTRACT
)CM       SPACE=(TRK,(300,1)),
)CM       DCB=(RECFM=FB,LRECL=1200,BLKSIZE=27600)
)CM -------------------------------------------
//&userid.8 JOB MSGCLASS=X,TIME=1440,REGION=4M,NOTIFY=&userid
//STEP00   EXEC PGM=IEFBR14
//FILEDEL  DD DSN=&userid..RMM.PRIV,
//         DISP=(MOD,DELETE,DELETE),
```

```
//         UNIT=SYSDA,SPACE=(TRK,(1,1))
/*
//STEPØ1   EXEC PGM=EDGHSKP,PARM='RPTEXT'
//SYSPRINT DD   SYSOUT=*
//MESSAGE  DD   DSN=RMM.MESSAGES,DISP=SHR
//REPTEXT  DD   DSN=RMM.REPORT.EXTRACT,DISP=SHR
//STEPØ2   EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=Ø
//SYSUT1   DD   DSN=RMM.MESSAGES,DISP=SHR
//SYSUT2   DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SORTVOL  EXEC PGM=SORT,REGION=6M
//SYSPRINT DD   SYSOUT=*
//SORTWKØ1 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ2 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ3 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ4 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTIN   DD   DISP=SHR,DSN=RMM.REPORT.EXTRACT
//SORTOUT  DD   DSN=&&SORTVOL,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(2ØØ,2Ø),RLSE),UNIT=SYSALLDA,
//         DCB=*.SORTIN
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
  SORT FIELDS=(9,6,CH,A)
  OMIT COND=(5,4,CH,NE,C'V   ')
/*
//SORTDSN  EXEC PGM=SORT,REGION=6M
//SYSPRINT DD   SYSOUT=*
//SORTWKØ1 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ2 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ3 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ4 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTIN   DD   DISP=SHR,DSN=RMM.REPORT.EXTRACT
//SORTOUT  DD   DSN=&&SORTDSN,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(2ØØ,2Ø),RLSE),UNIT=SYSALLDA,
//         DCB=*.SORTIN
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
  SORT FIELDS=(1Ø9,6,CH,A)
  OMIT COND=(5,4,CH,NE,C'D   ')
/*
//EXTEXTR  EXEC PGM=IKJEFTØ1,DYNAMNBR=99,REGION=4Ø96K
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//DSN      DD   DSN=&&SORTDSN,DISP=(OLD,PASS)
//VOL      DD   DSN=&&SORTVOL,DISP=(OLD,PASS)
//EXTEXTR  DD   DSN=&&EXTEXTR,DISP=(,PASS),
//         SPACE=(TRK,(2ØØ,2Ø),RLSE),UNIT=SYSALLDA,
//         DCB=(LRECL=12ØØ,BLKSIZE=6ØØØ,RECFM=FB)
//SYSTSIN  DD   *
```

```
 EX 'SYS1.SEDGEXE1(EDGRRPTR)'
/*
//SORTØ4   EXEC PGM=SORT,REGION=6M
//* ******************************************************************
//* * CREATING AN INVENTORY LIST SORTED BY DATA SET NAME              *
//* ******************************************************************
//SYSPRINT DD   SYSOUT=*
//SORTWKØ1 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ2 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ3 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ4 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTIN   DD   DISP=SHR,DSN=RMM.EXTENDED.REPORT.EXTRACT
//SORTOUT  DD   DSN=&&TEMPØ4,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(2ØØ,2Ø),RLSE),UNIT=SYSALLDA,
//         DCB=*.SORTIN
//SYSOUT   DD   SYSOUT=*
//*        ************************************************************
//*        * SORTED BY DATA SET NAME, CREATE DATE AND CREATE TIME    *
//*        * EXCLUDE ALL VOLUMES WITHOUT ANY DATA SET NAME (BLANK)   *
//*        ************************************************************
//SYSIN    DD   *
  SORT FIELDS=(8Ø5,44,CH,A,849,16,CH,A)
  OMIT COND=(8Ø5,44,CH,EQ,C' ')
/*
//SORT11A  EXEC PGM=SORT,REGION=6M
//* ******************************************************************
//* * MULTI VOLUME / MULTI DATA SET                                   *
//* ******************************************************************
//SORTWKØ1 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ2 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ3 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTWKØ4 DD   SPACE=(CYL,(1Ø,1Ø)),UNIT=SYSALLDA
//SORTIN   DD   DSN=&&EXTEXTR,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&TEMP11A,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(2ØØ,2Ø),RLSE),UNIT=SYSALLDA,
//         DCB=*.SORTIN
//SYSOUT   DD   SYSOUT=*
//*        ************************************************************
//*        * SORTED BY MULTI DSET MULTI VOLUME ID, VOLUME SEQUENCE,  *
//*        *           VOLUME SERIAL NUMBER AND DATA SET SEQUENCE    *
//*        * INCLUDE ONLY MULTI VOLUMES AND/OR MULTI FILE VOLUMES    *
//*        ************************************************************
//SYSIN    DD   *
  SORT FIELDS=(29,8,CH,A,236,16,CH,A,314,4,CH,A,911,4,CH,A)
  INCLUDE COND=(13Ø,4,CH,GT,C'   1',
             OR,11,12,CH,NE,C'           ')
/*
//EXTEXTRM EXEC PGM=IKJEFTØ1,DYNAMNBR=99,REGION=4Ø96K
//* ******************************************************************
//* * EXPAND EXTENDED EXTRACT FILE                                    *
```

```
//* ****************************************************************
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//EXTEXTRI DD   DSN=&&TEMP11A,DISP=(OLD,DELETE)
//EXTEXTRO DD   DSN=&&LCLEXTRM,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(200,20),RLSE),UNIT=SYSALLDA,
//         DCB=(RECFM=FB,LRECL=1244,BLKSIZE=22392)
//SYSTSIN  DD   *
 EX 'SYS1.SEDGEXE1(EDGRRPTM)'
/*
//SORT11B  EXEC PGM=SORT,REGION=6M
//* ****************************************************************
//* * MULTI VOLUME / MULTI DATA SET                                 *
//* ****************************************************************
//SORTWK01 DD   SPACE=(CYL,(10,10)),UNIT=SYSALLDA
//SORTWK02 DD   SPACE=(CYL,(10,10)),UNIT=SYSALLDA
//SORTWK03 DD   SPACE=(CYL,(10,10)),UNIT=SYSALLDA
//SORTWK04 DD   SPACE=(CYL,(10,10)),UNIT=SYSALLDA
//SORTIN   DD   DISP=(OLD,DELETE),DSN=&&LCLEXTRM
//SORTOUT  DD   DSN=&&TEMP11B,DISP=(,PASS,DELETE),
//         SPACE=(TRK,(200,20),RLSE),UNIT=SYSALLDA,
//         DCB=*.EXTEXTRM.EXTEXTRO
//SYSOUT   DD   SYSOUT=*
//*        ************************************************************
//*        * SORTED BY FIRST FILE ON FIRST VOLUME, MULTI DSET MULTI  *
//*        *           VOLUME ID, AL NUMBER AND DATASET SEQUENCE     *
//*        * ------------------------------------------------------- *
//*        * INCLUDE ALL RECORDS SELECTED IN STEP STEP11A            *
//*        ************************************************************
//SYSIN    DD   *
  SORT FIELDS=(1201,44,CH,A,29,8,CH,A,314,4,CH,A,911,4,CH,A)
/*
//EXTRPDT  EXEC PGM=IKJEFT01,DYNAMNBR=99,REGION=4096K
//OUTDDQ   OUTPUT FORMDEF=A10111, ** PRINT ON BOTH SIDES
//         PAGEDEF=V06483,        ** TEXT WILL BE ROTATED 90 DEGREES
//*                               ** AND PRINTED WITH 8 LINES P. INCH
//         CHARS=GT12,            ** FONT
//         DEST=NODE.ADDRESS
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SORT04   DD   DSN=&&TEMP04,DISP=(OLD,PASS)
//REPORT04 DD   SYSOUT=*,RECFM=VBA
//SORT11   DD   DSN=&&TEMP11B,DISP=(OLD,PASS)
//REPORT11 DD   DSN=&userid..RMM.PRIV,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(TRK,(30,1)),
//         DCB=(RECFM=VBA,LRECL=150,BLKSIZE=27998)
//SYSTSIN  DD   *
 EX 'SYS1.SEDGEXE1(EDGRRPTE)' -
    '054 N N N Y N N N N N N Y N +
     INTERNAL USE ONLY'
//
```

SQLISPF

```
//SYSADM1 JOB MSGCLASS=X,REGION=4M,NOTIFY=&SYSUID
//*************************************************************
//* Customized version of SQLSPDB published in the January   *
//*      1998 issue of DB2 Update                             *
//*                                                           *
//*************************************************************
//JOBLIB   DD  DISP=SHR,DSN=SYS1.DSN510.SDSNEXIT
//         DD  DISP=SHR,DSN=DSN510.SDSNLOAD
//PC      EXEC PGM=DSNHPC,PARM='HOST(ASM),SOURCE',REGION=4096K
//DBRMLIB  DD DSN=SYSADM.DBRMLIB5.DATA(SQLISPF),DISP=SHR
//STEPLIB  DD  DISP=SHR,DSN=SYS1.DSN510.SDSNEXIT
//         DD  DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSCIN   DD  DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(800,(10,10))
//SYSLIB DD    DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(800,(10,10),,,ROUND),UNIT=SYSDA
//SYSIN    DD  *
         TITLE ' REXX INTERFACE WITH DB2 - CAF'
/*********************************************************************
* FUNCTION : THE RESULTS FROM THE SPECIFIED ESQL QUERY ARE RETURNED   *
*            AS REXX VARIABLES.                                       *
*            THE VARIABLE NAMES ARE THOSE ATTRIBUTES RESULTING        *
*            FROM THE ESQL QUERY.                                     *
*            IF A NON-SELECT SQL STATEMENT IS ENTERED NO OUTPUT IS    *
*            RETURNED, BUT THE COMMAND IS EXECUTED.                   *
*            A MAXIMUM OF 9 FUNCTIONS (LIKE -SUM- OR -COUNT-) ARE     *
*            ALLOWED.                                                 *
*            IN CASE OF A NON-SELECT SQL COMMAND, YOU CAN HAVE ONE    *
*            (1) HOSTVARIABLE IN THE SQL STATEMENT (REPRESENTED BY A  *
*            ?). PUT THE VALUE OF THE HOSTVARIABLE IN REXX VARIABLE   *
*            HOSTVAL.                                                 *
*            A HOSTVARIABLE IN A SELECT SQL STATEMENT WILL RESULT IN  *
*            SQLCODE -313.                                            *
*            THIS PROGRAM USES THE DB2 CAF INTERFACE AND SO CAN       *
*            EXECUTE OUTSIDE THE DB2 ENVIRONMENT (HOWEVER, SQL MUST   *
*            BE AVAILABLE).                                           *
*            THIS PROGRAM IS WRITTEN FOR THE SP/DB DB2 AIDS ONLY.     *
*            NO GUARANTEES ARE MADE!!!!!!!!!                          *
*            THERE IS A KNOWN BUG IN THIS PROGRAM! DO NOT ATTEMPT     *
*            TO PERFORM ARITHMETIC FUNCTIONS LIKE SUM(COLUMN)/10 IN   *
*            THIS PROGRAM. IT WILL MESS UP THE PRECISION!             *
*            TRY AND FIX IT. IF YOU DO, MAIL IT TO ME!                *
* PLANNAME : SQLISPF                                                  *
* INPUT VARIABLE   : <DB2V>     : DB2 SYSTEM (DEFAULT IS DSN)         *
```

```
*                  : <SQLQUERY> : SPECIFIC SQL QUERY                    *
*                  : <HOSTVAL>  : OPTIONAL HOST VARIABLE                *
* OUTPUT VARIABLES : COLUMNNAME(J).I (I = ROW)                          *
*                    <_VN.>.J   : COLUMN NAMES                          *
*                    <_VN.Ø>    : # OF COLUMNS                          *
*                    <_NROWS>   : # OF SELECTED ROWS                    *
*                    <_VN1> THRU <_VN9> FOR EVERY FUNCTION THATS USED *
*                    <_REASON>  : REASONCODE (ONLY IF BAD CONNECT)      *
* RETURNCODES      : Ø    -   OK                                        *
*                    <N>  -   REXX OR SQL CODE                          *
*                    99   -   BAD CONNECT TO DB2                        *
************************************************************************
SQLISPF  CSECT
         REGEQ
         BEGIN SAVE=SA,BASE=(R11,R12)
         B     SA_END             jump over save-area
SA       DS    18A
SA_END   DS    ØH
         LOAD  EP=IRXEXCOM                LOAD IRXEXCOM
         ST    RØ,AIRXEXCOM               SAVE ENTRYPOINT
* OBTAIN DB2 SYSTEM
* INITIALIZE IT TO TEST DB2 (DSN)
* IF NO DB2V PARM, TAKE IT AS DEFAULT
         MVC   SSID,=C'DSN'
         LA    R5,IRX_SHVBLOCK
         USING SHVBLOCK,R5
         MVI   SHVCODE,SHVFETCH
         MVC   SHVBUFL,=A(L'DB2V)
         MVC   SHVVALA,=A(DB2V)
         MVC   VN,=C'DB2V '
         BAL   R14,GETVNL            GET LENGTH OF <VN>
         ST    RØ,SHVNAML            L(<VN>)
         MVC   SHVNAMA,=A(VN)        A(<VN>)
         L     R15,AIRXEXCOM
         CALL  (15),(IRX_IRXEXCOM,Ø,Ø,IRX_SHVBLOCK),VL
         LTR   R15,R15               REXX RETURN CODE
         BNZ   NODBPRM               PARAMETER MISSING, TAKE DEFAULT
         L     R1,SHVVALL            L(PARAMETER)
         STH   R1,DB2VAR
         MVC   SSID(4),DB2V
```

Editor's note: this article will be concluded in next month's issue of DB2 Update.

Nikola Lazovic and Gordana Kozovic
DB2 System Administrators
Postal Savings Bank (Yugoslavia)

DB2 news

Quest Software has released Version 1.1 of its Quest Central for DB2, with added support for OS/390 mainframes, DB2 Version 7, and new functions to help migrate databases across servers.

It lets DBAs perform tasks including performance monitoring, database administration, space management, and SQL tuning. The new release extends SQL tuning to OS/390 by providing a graphical tuning environment and context-sensitive advice to help improve database response times.

The software also facilitates the migration of databases, letting users drag a database or database object and drop it on to another server. It also migrates dependent objects, data, and security permissions.

It also supports new features found in DB2 Universal Database Version 7 and provides the ability to manage identity columns, SQL procedures, tablespace sizing parameters, and other new features.

For further information contact:
Quest Software, 8001 Irvine Center Drive, Irvine, CA 92618, USA.
Tel: (949) 754 8000.
URL: http://www.quest.com.

* * *

IBM has announced Version 7.2 of DB2 and Version 7.0 of DB2 UDB, with a range of enhancements focusing on business intelligence issues.

Version 7.2 includes data management enhancements, new assist wizards, an enhanced data warehouse centre with new sources including SAP R/3, i2 TradeMatrix BPI, Web clickstream data, OLE DB objects and MQSeries messages, including XML documents, and Sybase and Microsoft SQL Server wrappers to query and retrieve DB2 federated systems data.

The DB2 Universal Developer's Edition now includes a developer licence for Warehouse Manager, Data Links Manager, Net Search Extender, Relational Connect, and Spatial Extender. And UDB Text Information Extender is for searching text documents using an SQL query.

There's also a Warehouse Manager Sourcing Agent for OS/390, Intelligent Miner Scoring for deploying mining capabilities, and the DB2 Migration Toolkit for Sybase T-SQL.

IBM pointed out that it won't be releasing a DB2 UDB Satellite Edition Version 7, but in the next version of DB2 UDB that function will be merged into UDB Personal Edition.

Meanwhile, Version 7.1 of DB2 OLAP Server for OS/390 was announced, enabling bigger cubes to be created within a given batch window. The OLAP Integration Server runs natively on S/390, storage can be either multi-dimensional or relational, there are improved I/Os and larger files for scalability, and all OLAP Server Version 7.1 add-ons, including the new Allocations Manager and Analyzer, can be used with DB2 OLAP Server for OS/390.

For further information contact your local IBM representative.
URL: http://www.ibm.com.