In this issue

3  DB2 UDB LUW – checking whether a table is being accessed
6  SQL scalar functions
19  Stinger
21  DB2 object manager – part 2
50  DB2 news

© Xephon Inc 2004
**DB2 Update**

**Published by**
Xephon Inc  
PO Box 550547  
Dallas, Texas 75355  
USA  
Phone: 214-340-5690  
Fax: 214-341-7081

**Editor**
Trevor Eddolls  
E-mail: trevore@xephon.com

**Publisher**
Nicole Thomas  
E-mail: nicole@xephon.com

**Subscriptions and back-issues**
A year’s subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £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 2000 issue, are available separately to subscribers for $33.75 (£22.50) 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/db2; you will need to supply a word from the printed issue.

---

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

**Contributions**
When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

---

© Xephon Inc 2004. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

*Printed in England.*
DB2 UDB LUW – checking whether a table is being accessed

This article looks at the problem of determining whether a DB2 UDB for LUW table is being accessed or not. Have you ever been in the situation where you have a table on your production system but you are not sure whether any of your queries still access that table? You could always rename the table and see whether anyone complains (but remember there are restrictions on which tables you can and cannot rename – see UDB – restrictions on renaming a table in issue 123 of DB2 Update, January 2003). A better method would be to see whether we can find a method/tool to give us this information.

If we are talking about detecting SELECTs from a table then we cannot use ‘instead of’ triggers because they only work for INSERT/DELETE/UPDATE operations.

To detect SELECTs, you could run an EVENT monitor that looks at the SQL being run and checks for SELECT statements. This would incur an overhead on your system, and you may not be able to process the volume of output that is generated – it could be a complex and time-consuming process. Or you could use the snapshot feature to see the number of rows of each table accessed, which sounds easy in theory, but may be more difficult to implement in practice. But as this is the only ‘free’ method available that I can think of, let’s see what you would have to do.

I ran all the SQL in this article on a Windows 2000 machine running DB2 UDB 8.1 FP2 using the DB2ADMIN userid and the SAMPLE database.

First you have to activate the snapshot table monitoring switch. I did this by updating the dft_mon_table monitor switch at the instance level (and then stopping/startling the instance):

>db2 update dbm cfg using dft_mon_table on
(>db2stop then >db2start)

(You could of course have just issued a DB2 UPDATE MONITOR SWITCHES USING TABLE ON command to set the monitor switch on locally.)

Let me access the EMPLOYEE table in the SAMPLE database as follows:

>db2 connect to sample

>db2 select count(*) from employee

I can now issue a **get snapshot for tables** command to retrieve the required information (or if you are running DB2 UDB V8 you can use the snapshot table function shown later).

>db2 get snapshot for tables on sample

**Only part of the output is shown below:**

<table>
<thead>
<tr>
<th>Table List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Schema     = DB2ADMIN</td>
</tr>
<tr>
<td>Table Name       = EMPLOYEE</td>
</tr>
<tr>
<td>Table Type       = User</td>
</tr>
<tr>
<td>Rows Read        = 32</td>
</tr>
<tr>
<td>Rows Written     = Ø</td>
</tr>
<tr>
<td>Overflows        = Ø</td>
</tr>
<tr>
<td>Page Reorgs      = Ø</td>
</tr>
</tbody>
</table>

You can see that 32 rows were selected from the table db2admin.employee (which shows that the table was accessed!). If we retrieve the table schema and table name information from the snapshot command output and compare them with the tabschema and tabname columns from the syscat.tables catalog table, we can see which tables have not been accessed for the period between when the monitor was switched on and when the **get snapshot** command was issued. This is an important point – some tables may be accessed only at month/quarter/year end, so make sure that the monitor is run over a representative period!

As mentioned above, if we are running UDB DB2 V8.1 or above, we can use the snapshot table functions to obtain the table schema and name and the rows read using a query of the
form (I am limiting my search to tables with a schema of DB2ADMIN):

```
select
    substr(table_schema,1,20),
    substr(table_name,1,20),
    rows_read
from
    table(snapshot_table('sample',-1)) as s
where
    table_schema = 'DB2ADMIN'
order by rows_read desc
```

1                    2                    ROWS_READ
-------------------- -------------------- -------------------
DB2ADMIN             EMPLOYEE                               32

1 record(s) selected.

Now we can join this with the tabschema and tabname columns of the syscat.tables catalog to get a list of tables that have not been accessed. The query could look something like (again, limiting my search to tables having a schema of DB2ADMIN):

```
select
    substr(a.tabschema,1,20),
    substr(a.tabname,1,20)
from
    syscat.tables a
where
    (a.tabname not in (select  s.table_name from
        table(snapshot_table('sample',-1)) as s where s.table_schema = 'DB2ADMIN') and a.tabschema = 'DB2ADMIN' )
```

The output will look like:

```
1                    2
-------------------- -------------------
DB2ADMIN             CL_SCHED
DB2ADMIN             DEPARTMENT
DB2ADMIN             EMP_ACT
DB2ADMIN             EMP_PHOTO
DB2ADMIN             EMP_RESUME
DB2ADMIN             ORG
DB2ADMIN             PROJECT
DB2ADMIN             SALES
DB2ADMIN             STAFF
```

These are the tables from the SAMPLE database, excluding
the EMPLOYEE table. So now we have a list of tables which were not accessed in the period that we ran the monitor for.

You could of course run the select from the snapshot tables and populate a table with the information (perhaps together with a time stamp), which will then give you a historical perspective of which tables were accessed when.

So are there any tools available? IBM and other vendors have tools that will give you the information. Briefly, from IBM there are the Recovery Expert tool and the Query Patroller offering. If you have the Recovery Expert tool, one of the options is to check through the logs and produce a report of the number of times a table was accessed. This can be run offline, and you can specify how many logs to process etc. The Query Patroller offering also has a feature that reports on which tables have been accessed in a given period of time, and also which columns have been accessed.

I hope I have shown you a simple way to check whether a table has been read or not. This method requires you to write some queries and will involve manual effort in running and processing the information.

C Leonard
Freelance Consultant (UK) © Xephon 2004

SQL scalar functions

DB2 introduced in Version 7 SQL scalar functions, which provide a quick and easy way to write simple user-defined functions. You write code for an SQL scalar function when you define it, which eliminates the need to write and prepare a host-language program. In addition, because the source code for an SQL scalar function is stored in the DB2 catalog, an SQL scalar function performs better than an external user-defined function providing the same function.
An SQL scalar function is a user-defined function in which the CREATE FUNCTION statement contains the source code. The source code is a single SQL expression that evaluates to a single value. The SQL scalar function can return only one parameter. You specify the SQL expression in the RETURN clause of the CREATE FUNCTION statement. The value of the SQL expression must be compatible with the data type of the parameter in the RETURN clause. The body of an SQL function must not contain a recursive call to itself or to another function or method that calls it. Similar to views, the body text is stored in the catalog in SYSIBM.SYSVIEWS (TYPE='F') and merged into the query at bind time.

A user-defined function is a mechanism by which you can write your own extensions to the SQL language. The built-in functions supplied with DB2 may not satisfy all your requirements. There are several reasons why we deployed SQL scalar UDFs on the mainframe. These include a greater compatibility with the DB2 product family, and a greater capability to port applications from other DBMSs to DB2. Many of the programs at your site implement the same basic set of functions, but there are minor differences in all the implementations—so, you are unsure about the consistency of the results you receive. If you correctly implement these functions once, in a UDF, then all these programs can use the same implementation directly in SQL and provide consistent results. With a UDF you can encapsulate the logic of having to write a complex expression into a UDF. Replacing a complex expression by a UDF improves readability of the SQL statement. It can also avoid coding errors because you can easily make a mistake when repeatedly coding the same complex expressions. SQL scalar UDFs also offer some advantages over external UDFs in that they consume significantly fewer resources, and don’t require any WLM (Workload Manager) application environment set-up.

I have created ten new SQL and external user-defined functions. The functions and their type are shown below:
CENTER

The CENTER(host variable, length, pad) function returns a string of length \textit{length} with the string centred in it. Pad characters are added as necessary to make up \textit{length}. The default pad character is blank. If the string is longer than \textit{length}, it will be truncated at both ends to fit.

Here are some examples:

\begin{align*}
\text{CENTER('XYZ', 7)} & \quad \text{' XYZ '}
\text{CENTER('XYZ', 8, '-')} & \quad \text{'--XYZ--'}
\text{CENTER('The Lord of the Rings', 16)} & \quad \text{e Lord of the Ri}'
\end{align*}

Code:

\begin{verbatim}
CREATE FUNCTION SYSADM.CENTER
(ITEM VARCHAR(1024), NLEN INTEGER)
RETURNS VARCHAR(1024)
SPECIFIC CENTER LANGUAGE SQL
RETURN
CASE WHEN LENGTH(STRIP(ITEM)) = NLEN THEN ITEM
     WHEN LENGTH(STRIP(ITEM)) < NLEN THEN REPEAT(' ',SMALLINT((NLEN-LENGTH(STRIP(ITEM))
          )/FLOAT(2)))||ITEM||
             REPEAT(' ',SMALLINT(CEIL((NLEN-LENGTH(STRIP(ITEM))
          )))
     ELSE ITEM END
\end{verbatim}
WHEN LENGTH(STRIP(ITEM)) > NLEN
    THEN SUBSTR(ITEM,SMLINT((LENGTH(STRIP(ITEM))-NLEN)/FLOAT(2))+1,NLEN) ELSE ''
END;

CREATE FUNCTION SYSADM.CENTER
    (ITEM VARCHAR(1024), NLEN INTEGER, CHR VARCHAR(1))
RETURNS VARCHAR(1024)
SPECIFIC CENTERC LANGUAGE SQL
RETURN
CASE WHEN LENGTH(STRIP(ITEM)) = NLEN THEN ITEM
    WHEN LENGTH(STRIP(ITEM)) < NLEN
        THEN REPEAT(CHR,SMLINT((NLEN-LENGTH(STRIP(ITEM)))/FLOAT(2)))||ITEM||
            REPEAT(CHR,SMLINT(CEIL((NLEN-LENGTH(STRIP(ITEM)))/FLOAT(2))))
    WHEN LENGTH(STRIP(ITEM)) > NLEN
        THEN SUBSTR(ITEM,SMLINT((LENGTH(STRIP(ITEM))-NLEN)/FLOAT(2))+1,NLEN) ELSE ''
END;

DTYPE

The DTYPE(host variable) function returns a value ‘N’ if the input host variable or string contains only numeric characters in the range 0–9. If the string contains any non-numeric characters, then the function returns the value ‘A’. I have also defined a DTYPE SQL function on a local DB2 UDB 7.2.

Example:

SELECT SYSADM.DTYPE('911') A,
    SYSADM.DTYPE('91X') B
FROM SYSIBM.SYSDUMMY1
WITH UR;

A    B
--    ---
N    A

Code for mainframe DB2:

    /*! Mainframe DB2 for z/OS */
    CREATE FUNCTION SYSADM.DTYPE
        ( ITEM VARCHAR(255) )
    RETURNS CHAR(1)
    SPECIFIC DTYPE
EXTERNAL NAME 'DTYPE'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID DTYPE
WLM ENVIRONMENT DSNNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
NO SCRATCHPAD
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2 ;

Code for DB2 UDB:

/* Local DB2 UDB 7.2          */
CREATE FUNCTION NADI.DTYPE (ITEM VARCHAR(255))
RETURNS  CHAR(1)
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN
WITH DT (I, ITEM, ZNAK) AS
( SELECT Ø, ITEM, ''
     FROM SYSIBM.SYSDUMMY1
     UNION ALL
     SELECT I+1, ITEM,
          SUBSTR(ITEM,I+1,1)
     FROM DT
     WHERE I < LENGTH(ITEM))
SELECT CASE
     WHEN SNUM=1Ø THEN 'N' ELSE 'A'
END
FROM (SELECT SUM(CASE
     WHEN NUM=Ø THEN 1ØØ ELSE 1Ø
     END ) SNUM
     FROM (SELECT DISTINCT POSSTR(ZNAK,'Ø')+POSSTR(ZNAK,'1')+POSSTR(ZNAK,'2')+
     POSSTR(ZNAK,'3')+POSSTR(ZNAK,'4')+POSSTR(ZNAK,'5')+POSSTR(ZNAK,'6')+
     POSSTR(ZNAK,'7')+POSSTR(ZNAK,'8')+POSSTR(ZNAK,'9') NUM
     FROM DT
     WHERE ZNAK <> '' ) X ) Y
**DELWORD**

The DELWORD(host variable or string, n) function deletes the substrings of *string* that starts at the *n*th word. Number *n* must be a positive whole number. If *n* is greater than the number of words in *string*, *string* is returned unchanged.

Example:

```
DELWORD('The Lord of the Rings', 1)       ->      'Lord of the Rings'
```

Code:

```
CREATE FUNCTION SYSADM.DELWORD
    ( ITEM VARCHAR(2000)
      , NTH  INTEGER )
RETURNS  VARCHAR(2000)
SPECIFIC DELWORD
EXTERNAL NAME 'DELWORD'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID DELWORD
WLM ENVIRONMENT DSNNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2 ;
```

**JRIGHT**

The JRIGHT(host variable) function returns a small integer or integer host variable in character format with leading blank characters.

Example:

```
SELECT INFO.JRIGHT(smallint(707)) small,
       INFO.JRIGHT(707) integer
FROM SYSIBM.SYSDUMMY1
WITH UR
```
Result:

<table>
<thead>
<tr>
<th>SMALL</th>
<th>INTEGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>'7Ø7'</td>
<td>'7Ø7'</td>
</tr>
</tbody>
</table>

Code:

CREATE FUNCTION INFO.JRIGHT
(ITEM SMALLINT)
RETURNS VARCHAR(31)
LANGUAGE SQL
SPECIFIC JRIGHTS
RETURN REPEAT(' ',LENGTH(CHAR(ITEM)) -
LENGTH(STRIP(CHAR(ITEM))))
CONCAT STRIP(CHAR(ITEM));

CREATE FUNCTION INFO.JRIGHT
(ITEM INTEGER)
RETURNS VARCHAR(31)
LANGUAGE SQL
SPECIFIC JRIGHTI
RETURN REPEAT(' ',LENGTH(CHAR(ITEM)) -
LENGTH(STRIP(CHAR(ITEM))))
CONCAT STRIP(CHAR(ITEM));

LASTPOS
The LASTPOS(input_string, search_string) function returns the position of the last occurrence of a search string in input_string. If the search string is not found, 0 is returned. The LASTPOS function uses also the REVERSE user-defined function, which has been published in DB2 Update, issues 103 and 104, May and June 2001, in ‘Sample user-defined-functions’.

Here are some examples:

LASTPOS('The Lord of the Rings', 'the')  -> 13
LASTPOS('The Lord of the Rings', 'The')  -> 1
LASTPOS('The Lord of the Rings', 'THE')  -> 0

Code:

CREATE FUNCTION SYSADM.LASTPOS
(ITEM VARCHAR(1000), SITEM VARCHAR(500))
RETURNS INTEGER
SPECIFIC LASTPOS
SUBWORD

The SUBWORD(host variable or string, n) function returns the substrings of string that starts at the nth word. Number n must be a positive whole number. If n is greater than the number of words in string, string is returned with blank characters.

Example:

```
SUBWORD('The Lord of the Rings', 4)       ->      'the Rings'
```

Code:

```
CREATE FUNCTION SYSADM.SUBWORD
    ( ITEM VARCHAR(2000)
        , NTH  INTEGER )
RETURNS  VARCHAR(2000)
SPECIFIC SUBWORD
EXTERNAL NAME 'SUBWORD'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID SUBWORD
WLM ENVIRONMENT DSNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2 ;
```
**WORD**

The WORD(host variable or string, n) function returns the \( n \)th blank-delimited word in *string*. The parameter \( n \) must be a positive whole number. If there are fewer than \( n \) words in *string*, the blank string is returned.

Here are some examples:

```
WORDS('The Lord of the Rings', 2)            ->  'Lord'
WORDS('The Lord of the Rings', 6)            ->  '
```

**Code:**

```
CREATE FUNCTION SYSADM.WORD
    ( ITEM VARCHAR(2000)
    , NTH  INTEGER )
RETURNS  VARCHAR(2000)
SPECIFIC WORD
EXTERNAL NAME 'WORD'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
NO SQL
DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2 ;
```

**WORDINDEX**

The WORDINDEX(host variable or string, n) function returns the position of the first character in the \( n \)th blank-delimited word in *string*. The parameter \( n \) must be a positive whole number. If there are fewer than \( n \) words in *string*, 0 is returned.

Here are some examples:

```
WORDINDEX('The Lord of the Rings', 2)            ->  5
WORDINDEX('The Lord of the Rings', 6)            ->  0
```
Code:

CREATE FUNCTION SYSADM.WORDINDEX
  ( ITEM VARCHAR(2000)
    , NTH INTEGER )
RETURNS INTEGER
SPECIFIC WORDINDEX
EXTERNAL NAME 'WORDINDX'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID WORDINDX
WLM ENVIRONMENT DSNNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2 ;

WORDLENGTH

The WORDLENGTH(host variable or string, n) function returns the length of the \( n \)th blank-delimited word in \textit{string}. The parameter \( n \) must be a positive whole number. If there are fewer than \( n \) words in \textit{string}, 0 is returned.

Here are some examples:

\[
\text{WORDLENGTH('The Lord of the Rings', 2)} \rightarrow 4 \\
\text{WORDLENGTH('The Lord of the Rings', 6)} \rightarrow 0
\]

Code:

CREATE FUNCTION SYSADM.WORDLENGTH
  (ITEM VARCHAR(2000), NWORD INTEGER)
RETURNS INTEGER
SPECIFIC WORDLENGTH
LANGUAGE SQL
RETURN
CASE WHEN NWORD > SYSADM.WORDS(ITEM)
  OR NWORD = 0 THEN 0
WORDS

The WORDS(host variable or string) function returns the number of blank-delimited words in string.

Here are some examples:

WORDS('The Lord of the Rings') -> 5
WORDS(' ') -> Ø

Code:

CREATE FUNCTION SYSADM.WORDS
  ( ITEM VARCHAR(2000) )
RETURNS INTEGER
SPECIFIC WORDS
EXTERNAL NAME 'WORDS'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
NO SQL
DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNWLM1
STAY RESIDENT YES
PROGRAM TYPE SUB
EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

PL/I SOURCE CODE FOR EXTERNAL FUNCTIONS

DTYPE

* PROCESS SYSTEM(MVS);
DTYPE: PROC(UDF_PARM1, UDF_RESULT,
    UDF_IND1, UDF_INDR,
    UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
    UDF_DIAG_MSG, UDF_SCRATCHPAD,
    UDF_CALL_TYPE, UDF_DBINFO)
OPTIONS(FETCHABLE NOEXECOPS REENTRANT);

/*****************************/
/* UDF : DTYPE */
/* INPUT : UDF_PARM1 VARCHAR(255) */
/* OUTPUT: UDF_RESULT VARCHAR(1) */
/*****************************/

DCL UDF_PARM1 CHAR(255) VAR;  /* INPUT PARAMETER */
DCL UDF_RESULT CHAR(1);  /* RESULT PARAMETER */
DCL UDF_IND1 BIN FIXED(15);  /* INDICATOR FOR INPUT PARM */
DCL UDF_INDR BIN FIXED(15);  /* INDICATOR FOR RESULT */
DCL (I,X,LEN) BIN FIXED(15);  /* INDICATOR FOR RESULT */

DCL 1 UDF_SCRATCHPAD,  /* SCRATCHPAD */
3 UDF_SPAD_LEN BIN FIXED(31),
3 UDF_SPAD_TEXT CHAR(100);

EXEC SQL INCLUDE UDFINFO;  /* DBINFO */

DCL (ADDR,LENGTH,SUBSTR,NULL,INDEX) BUILTIN;
EXEC SQL INCLUDE SQLCA;

UDF_RESULT='A';
EXEC SQL SET :UDF_PARM1=STRIP(:UDF_PARM1,B);
LEN=LENGTH(UDF_PARM1);
IF LEN>Ø THEN DO;
X=1;
DO I=1 TO LEN WHILE (X¬=Ø);
X=INDEX('Ø123456789',SUBSTR(UDF_PARM1,I,1));
END;
IF X=Ø THEN UDF_RESULT='A';
ELSE UDF_RESULT='N';
END;
END DTYPE;

DELWORD

* PROCESS SYSTEM(MVS);
DELW1: PROC(UDF_PARM1, UDF_PARM2, UDF_RESULT,
       UDF_IND1,  UDF_INDR,
       UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
       UDF_DIAG_MSG, UDF_SCRATCHPAD,
       UDF_CALL_TYPE, UDF_DBINFO)
       OPTIONS(FETCHABLE NOEXECOPS REENTRANT);

/*****************************/
/* UDF : DELWORD */
/* INPUT : UDF_PARM1 VARCHAR INPUT STRING */
/* OUTPUT: UDF_RESULT VARCHAR */
/*****************************/

DCL UDF_PARM1 CHAR(2000) VAR;  /* INPUT PARAMETER */
DCL UDF_PARM2 BIN FIXED(31);  /* INPUT PARAMETER */
DCL UDF_RESULT CHAR(2000) VAR;  /* RESULT PARAMETER */
DCL NWORDS BIN FIXED(15);  /* SEARCH STRING */
DCL UDF_IND1 BIN FIXED(15); /* INDICATOR FOR INPUT PARM */
DCL UDF_IND2 BIN FIXED(15); /* INDICATOR FOR INPUT PARM */
DCL UDF_INDR BIN FIXED(15); /* INDICATOR FOR RESULT */
DCL 1 UDF_SCRATCHPAD,/* SCRATCHPAD */
    3 UDF_SPAD_LEN BIN FIXED(31),
    3 UDF_SPAD_TEXT CHAR(100);
EXEC SQL INCLUDE SQLCA;
INCLUDE UDFINFO; /* DBINFO */
DCL (LENGTH,SUBSTR,ADDR,NULL) BUILTIN;
EXEC SQL SET :NWORDS=SYSADM.WORDS(:UDF_PARM1);
IF UDF_PARM2 > NWORDS | UDF_PARM2 < 1
THEN UDF_RESULT=UDF_PARM1;
ELSE DO;
    EXEC SQL SET :UDF_RESULT = REPLACE(:UDF_PARM1||' ','
    SYSADM.WORD(:UDF_PARM1,:UDF_PARM2)||'',''');
    IF SQLCODE¬=Ø THEN UDF_RESULT=' ';
END;
END DELW1;

SUBWORD

* PROCESS SYSTEM(MVS);
SUBW1: PROC(UDF_PARM1, UDF_PARM2, UDF_RESULT,
    UDF_IND1, UDF_INDR,
    UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
    UDF_DIAG_MSG, UDF_SCRATCHPAD,
    UDF_CALL_TYPE, UDF_DBINFO)
    OPTIONS(FETCHABLE NOEXECOPS REENTRANT);

/**************************************************************************/
/*    UDF   : SUBWORD                                               */
/*    INPUT : UDF_PARM1    VARCHAR  INPUT STRING                    */
/*    INPUT : UDF_PARM2    INTEGER                                  */
/*    OUTPUT: UDF_RESULT   VARCHAR  SUBWORD                         */
/**************************************************************************/
DCL UDF_PARM1 CHAR(2000) VAR; /* INPUT PARAMETER */
DCL UDF_PARM2 BIN FIXED(31); /* INPUT PARAMETER */
DCL UDF_RESULT CHAR(2000) VAR; /* RESULT PARAMETER */
DCL SSTR CHAR(2000) VAR; /* SEARCH STRING */
DCL PPOS BIN FIXED(15); /* SEARCH STRING */
DCL UDF_IND1 BIN FIXED(15); /* INDICATOR FOR INPUT PARM */
DCL UDF_IND2 BIN FIXED(15); /* INDICATOR FOR INPUT PARM */
DCL UDF_INDR BIN FIXED(15); /* INDICATOR FOR RESULT */
DCL 1 UDF_SCRATCHPAD,/* SCRATCHPAD */
    3 UDF_SPAD_LEN BIN FIXED(31),
    3 UDF_SPAD_TEXT CHAR(100);
EXEC SQL INCLUDE SQLCA;
INCLUDE UDFINFO; /* DBINFO */
DCL (LENGTH,SUBSTR,ADDR,NULL) BUILTIN;
EXEC SQL SET :PPOS = SYSADM.WORDINDEX(:UDF_PARM1,:UDF_PARM2);
Stinger

DB2 Version 8.1 (which shipped in November 2002) isn’t the cutting edge any more. IBM has posted a beta copy of the next release of its DB2 UDB for Linux, Unix, and Windows, code-named Stinger.

It boasts new query optimization technology, called a learning optimizer (LEO, from IBM Research), which automatically and continually updates query statistics about how the database is being used and how it is performing. It looks dynamically at query results and how the query interacted with the data. It is designed to learn from each interaction, so that next time the query will run faster.

There are more ‘autonomic’ (self-managing) features, which are designed to reduce the time spent on both routine and complex database maintenance tasks. IBM is also providing new DB2 Design Advisor features that automatically maintain, configure, deploy, and optimize the database performance, and allow certain tasks to be completed 6.5 times faster than if they required human intervention, IBM claims. An automatic object maintenance feature performs administration and maintenance functions, such as table adjustments or data back-ups.

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

Bernard Zver (bernard.zver@informatika.si)
DBA
Informatica (Slovenia)
A client rerouting capability enables a user’s desktop to automatically fail over to a back-up database when the primary DB2 database server has gone off-line. This could ensure DB2 availability during both planned and unplanned downtime.

The new tools allow developers to use either the Microsoft Visual Studio .NET tool set or Rational XDE Developer to design databases and database applications. Programmers will also be able to exploit the native .NET Data Provider, strengthening the .NET connection between databases and applications. DB2 has been added into palettes of developers for Visual Studio and Eclipse. If they point that Visual Studio palette at DB2, more features come along. They could write stored procedures in CLR (Microsoft’s Common-Language Run-time) and get more context-sensitive help.

The new DB2 features are a direct result of IBM’s acquisition of Rational Software. There is also extensive support for IBM’s WebSphere Studio frameworks.

The product also supports the new Version 2.6 of the Linux kernel. This helps database clusters scale higher and perform faster as well as better exploit the speed of 64-bit databases and servers that rely on multiple processors. IBM says that such multiprocessor servers can be joined in Linux clusters, as with DB2 ICE (Integrated Cluster Environment), an integrated package that combines DB2 and eServer Linux Cluster 1350 (xSeries, 325, BladeCenter) to provide a solution that, according to IBM, can cluster from two to 1,000 servers and pick up nodes at the rate of four per hour. New clustering features also automatically partition and optimize large databases on many servers, in just a few minutes rather than hours.

IBM has also announced an open beta of the next version of DB2 Information Integrator software, code-named Masala. The new software delivers over 100 new features focused on automation, faster access to relevant corporate data, simplified application deployment, and integration across the broadest array of information assets. The software enables users to access all types of data as though stored in one location.
DB2 object manager – part 2

This month we conclude the code that will provide a recommendation list for extent, image copy, reorg, restrict, and runstats, based on the user input criteria.

4300-GET-RESULT2.
   EXEC SQL FETCH C2 INTO :RS-OUTPUT-2 :RS-OUT2-IND
   END-EXEC.
   MOVE 'FETCH' TO DB2-COMMAND.
   PERFORM 9000-CHECK-SQLCODE.
   MOVE RS-DBNAME TO EX-DBNAME(I).
   MOVE RS-NAME TO EX-NAME(I).
   MOVE RS-OBJECTTYPE TO EX-OBJECTTYPE(I).
   MOVE RS-TOTALEXTENTS TO EX-TOTALEXTENTS(I).
   MOVE RS-ASSOCIATEDTS TO EX-ASSOCIATEDTS(I).
   IF RS-OUT2-IND(19) NOT < Ø THEN
     MOVE RS-REORGLASTTIME TO EX-REORGLASTTIME(I)
   ELSE
     GO TO 4300-EXIT.
   IF RS-OUT2-IND(20) < Ø OR RS-RRTINSDELUDPCT EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-RRIDU-II(I)
   ELSE
     MOVE RS-RRTINSDELUDPCT TO EX-RRTINSDELUDPCT(I).
   IF RS-OUT2-IND(21) < Ø OR RS-RRTUNCINSPECT EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-RRUCI-II(I)
   ELSE
     MOVE RS-RRTUNCINSPECT TO EX-RRTUNCINSPECT(I).
   IF RS-OUT2-IND(22) < Ø OR RS-RRTDISORGLOBPCT EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-RRDOL-II(I)
   ELSE
     MOVE RS-RRTDISORGLOBPCT TO EX-RRTDISORGLOBPCT(I).
   IF RS-OUT2-IND(23) < Ø OR RS-RRTMASSDELETE EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-RRMSD-II(I)
   ELSE
     MOVE RS-RRTMASSDELETE TO EX-RRTMASSDELETE(I).
   IF RS-OUT2-IND(24) < Ø OR RS-RRTINDREF EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-RRIDR-II(I)
   ELSE
     MOVE RS-RRTINDREF TO EX-RRTINDREF(I).
   IF RS-PARTITION EQUAL TO Ø THEN
     MOVE LOW-VALUES TO EX-PAR-II(I)
   ELSE
     MOVE RS-PARTITION TO EX-PARTITION(I).
   DISPLAY LIST-REORG-DEF(I).
   COMPUTE J = J + 1.
4300-EXIT.
EXIT.
9000-CHECK-SQLCODE.
*****************************************************************************
* VERIFY THAT THE PRIOR SQL CALL COMPLETED SUCCESSFULLY
*****************************************************************************
IF SQLCODE NOT = 0 AND SQLCODE NOT = 100 THEN
  MOVE 'BAD' TO RUN-STATUS
  DISPLAY '*    UNEXPECTED SQLCODE FROM SYSPROC.DANACCOR'
  ' DURING ' DB2-COMMAND ' REQUEST.'
  DISPLAY '*'
  PERFORM 9100-DETAIL-SQL-ERROR.
9100-DETAIL-SQL-ERROR.
*****************************************************************************
* CALL DSNTIAR TO RETURN A TEXT MESSAGE FOR AN UNEXPECTED
* SQLCODE.
*****************************************************************************
  CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
  IF RETURN-CODE = ZERO
    PERFORM 9200-PRINT-SQL-ERROR-MSG VARYING ERROR-INDEX
      FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 10.
  9200-PRINT-SQL-ERROR-MSG.
*****************************************************************************
* PRINT MESSAGE TEXT
*****************************************************************************
  DISPLAY ERROR-TEXT (ERROR-INDEX).

RESTRIND COBOL

IDENTIFICATION DIVISION.
PROGRAM-ID.  RESTRIND.
AUTHOR.      LIJUN GAO;
DATE-WRITTEN. 08/08/03.
DATE-COMPILED.
*****************************************************************************
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
*****************************************************************************
DATA DIVISION.
FILE SECTION.
*****************************************************************************
WORKING-STORAGE SECTION.
*****************************************************************************
* DISPLAY FIELDS FOR INPUT CRITERIA
*****************************************************************************
Ø1 DIS-EXTENTLIMIT PIC ZZZ9.
**OUTPUT TITLE FOR OBJECTS EXCEED RESTRICT CRITERIA LIMITS**

**OUTPUT LIST FOR OBJECTS EXCEED RESTRICT LIMITS**

**COPY ALL RELATED WORKING STORAGE DEFINITION**

**DB2 AREA**
PROCEDURE DIVISION USING RESTRICT-REC.

0000-MAIN-LOGIC.

PERFORM 1000-INIT THRU 1000-EXIT.
PERFORM 2100-PROCESS-PARMS THRU 2100-EXIT.
PERFORM 2200-PROCESS-PARMS THRU 2200-EXIT.
PERFORM 3000-CONNECT-TO-SERVER THRU 3000-EXIT.
IF OKAY THEN
   PERFORM 4000-CALL-DSNACCOR THRU 4000-EXIT
ELSE
   DISPLAY 'CONNECT NOT SUCCESSFUL'
   MOVE 8 TO RETURN-CODE.
EXEC SQL
   CONNECT RESET
END-EXEC.
STOP RUN.

1000-INIT.
MOVE 'GOOD' TO RUN-STATUS.
ACCEPT REFMOD-TIME-ITEM FROM TIME.
ACCEPT YYYYMMDD FROM DATE.
DISPLAY ".DSNG001I Job execution starting at "
   YYYYMMDD (5:2)
   "/"
   YYYYMMDD (7:2)
   "/2"
   YYYYMMDD (2:3)
   "  "
   REFMOD-TIME-ITEM (1:2)
   ":"
   REFMOD-TIME-ITEM (3:2)
   ":"
   REFMOD-TIME-ITEM (5:2)
   " ...".
DISPLAY ".DSNG001I MVS=SP7.0.3,PID=HBB7706,DFSMS=1.3.0'
   ',DB2=7.1.0'.
DISPLAY ".DSNG018I Connected to Subsystem ' DB2-LOC-NAME.
1000-EXIT.
EXIT.

2100-PROCESS-PARMS.
EVALUATE RESTRICT-OBJECT
WHEN "TYPE"
   MOVE RESTRICT-OBJECT-TYPE TO OBJECTTYPE-DTA
   MOVE 3 TO OBJECTTYPE-LN
WHEN OTHER
   DISPLAY ".DSNG013E Invalid Keyword ' RESTRICT-OBJECT
STOP RUN
END-EVALUATE
MOVE RESTRICT-DBNAME-VALUE TO CRI-VALUE.
MOVE 'RESTRICT' TO QUERYTYPE-DTA
MOVE 8 TO QUERYTYPE-LN
STRING
  CRI-NAME SPACE CRI-POINT
  DELIMITED BY SIZE
  CRI-VALUE
  DELIMITED BY SPACES
  CRI-POINT
  DELIMITED BY SIZE
  INTO CRITERIA-DTA.
MOVE 50 TO CRITERIA-LN.
2100-EXIT.
EXIT.
2200-PROCESS-PARMS.
****************************************************************
* PROCESS DSNACCOR INVOCATION PARAMETERS
****************************************************************
MOVE 59 TO CHKLVL.
DISPLAY ".DSNGØ15I QueryType = " QUERYTYPE
DISPLAY ".DSNGØ15I ObjectType = " OBJECTTYPE
DISPLAY ".DSNGØ15I QueryScope = WHERE " CRITERIA-DTA.
DISPLAY ".DSNGØ15I ".
DISPLAY ' '.
*******************************************************
* INITIALIZE OUTPUT PARAMETERS *
*******************************************************
MOVE SPACES TO LASTSTATEMENT-DTA.
MOVE 1 TO LASTSTATEMENT-LN.
MOVE 0 TO RETURNCODE.
MOVE SPACES TO ERRORMSG-DTA.
MOVE 1 TO ERRORMSG-LN.
MOVE 0 TO IFCARETCODE.
MOVE 0 TO IFCARESCODE.
MOVE 0 TO XSBYTES.
*******************************************************
* SET THE INDICATOR VARIABLES TO 0 FOR NON-NULL INPUT *
* PARAMETERS (PARAMETERS FOR WHICH YOU DO NOT WANT   *
* DSNACCOR TO USE DEFAULT VALUES) AND FOR OUTPUT      *
* PARAMETERS.                                        *
*******************************************************
MOVE 0 TO QUERYTYPE-IND.
MOVE 0 TO CHKLVL-IND.
MOVE 0 TO CRITERIA-IND.
MOVE 0 TO RRTINSDELUPDPCT-IND.
MOVE 0 TO RRTUNCLUSTINSPECT-IND.
MOVE 0 TO RRTDISORGLOBPCPT-IND.
MOVE 0 TO RRTMASSDELLIMIT-IND.
MOVE 0 TO EXTENTLIMIT-IND.
MOVE Ø TO LASTSTATEMENT-IND.
MOVE Ø TO RETURNCODE-IND.
MOVE Ø TO ERRORMSG-IND.
MOVE Ø TO IFCARETCODE-IND.
MOVE Ø TO IFCARESCODE-IND.
MOVE Ø TO XSBYTES-IND.

2200-EXIT.
EXIT.

3000-CONNECT-TO-SERVER.
****************************************************************
* CONNECT TO THE REMOTE SERVER
****************************************************************
EXEC SQL CONNECT TO :DB2-LOC-NAME END-EXEC.
MOVE 'CONNECT' TO DB2-COMMAND.
IF SQLCODE IS NOT EQUAL TO ZERO THEN
PERFORM 9000-CHECK-SQLCODE.

3000-EXIT.
EXIT.

4000-CALL-DSNACCOR.
*****************
* CALL DSNACCOR *
*****************
EXEC SQL CALL DSNACCOR
(:QUERYTYPE :QUERYTYPE-IND,
 :OBJECTTYPE :OBJECTTYPE-IND,
 :ICTYPE :ICTYPE-IND,
 :STATSSCHEMA :STATSSCHEMA-IND,
 :CATLGSHEMA :CATLGSHEMA-IND,
 :LOCALSCHEMA :LOCALSCHEMA-IND,
 :CHKLVL :CHKLVL-IND,
 :CRITERIA :CRITERIA-IND,
 :RESTRICTED :RESTRICTED-IND,
 :CRUPDATEDPAGESPCT :CRUPDATEDPAGESPCT-IND,
 :CRCANGESPCT :CRCANGESPCT-IND,
 :CRDAYSNCLASTCOPY :CRDAYSNCLASTCOPY-IND,
 :ICRUPDATEDPAGESPCT :ICRUPDATEDPAGESPCT-IND,
 :ICRANGESPCT :ICRANGESPCT-IND,
 :CRINDEXSIZE :CRINDEXSIZE-IND,
 :RRTINSDELUPDPCT :RRTINSDELUPDPCT-IND,
 :RRTUNCLUSTINSPECT :RRTUNCLUSTINSPECT-IND,
 :RRTDISORGLOBPCT :RRTDISORGLOBPCT-IND,
 :RRTMASSDELLIMIT :RRTMASSDELLIMIT-IND,
 :RRTINDREFLIMIT :RRTINDREFLIMIT-IND,
 :RRINSERTDELETEPCT :RRINSERTDELETEPCT-IND,
 :RRIAPPENDINSERTTPCT :RRIAPPENDINSERTTPCT-IND,
 :RRIAPPENDINSERTTPCT :RRIAPPENDINSERTTPCT-IND,
 :RRIPSEUDODELETEPCT :RRIPSEUDODELETEPCT-IND,
 :RRIMASSDELLIMIT :RRIMASSDELLIMIT-IND,
 :RRILEAFLIMIT :RRILEAFLIMIT-IND,
 :RRINUMLEVELSLIMIT :RRINUMLEVELSLIMIT-IND,
 :SRTINSDELUPDPCT :SRTINSDELUPDPCT-IND,
EXEC SQL ASSOCIATE LOCATORS(:LOC1, :LOC2) WITH PROCEDURE DSNACCOR
END-EXEC.
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :LOC1
END-EXEC
EXEC SQL ALLOCATE C2 CURSOR FOR RESULT SET :LOC2
END-EXEC
PERFORM 4050-DIS-TITLE
PERFORM 4300-GET-RESULT2 VARYING I FROM 1 BY 1 UNTIL SQLCODE EQUAL TO +100.
DISPLAY '----------------------------------------------------------'
'----------'
COMPUTE I = I - 1.
MOVE I TO DIS-I.
DISPLAY '.DSNGØ21I TOTAL ' DIS-I ' RECORDS RETRIEVED.'.
4050-DIS-TITLE.
DISPLAY '.
DISPLAY LIST-RESTRICT-NAMES.
DISPLAY '----------------------------------------------------------'
'----------'.
4300-GET-RESULT2.
EXEC SQL FETCH C2 INTO :RS-OUTPUT-2 :RS-OUT2-IND
END-EXEC.
MOVE 'FETCH' TO DB2-COMMAND.
PERFORM 9000-CHECK-SQLCODE.
MOVE RS-DBNAME TO EX-DBNAME(I).
MOVE RS-NAME TO EX-NAME(I).
MOVE RS-OBJECTTYPE TO EX-OBJECTTYPE(I).
IF RS-OUT2-IND(5) NOT < Ø THEN
  MOVE RS-OBJECTSTATUS TO EX-STATUS(I).
IF RS-PARTITION EQUAL TO Ø THEN
  MOVE LOW-VALUES TO EX-PAR-II(I)
ELSE
  MOVE RS-PARTITION TO EX-PARTITION(I).
END-EXEC.

****************************************************************
* VERIFY THAT THE PRIOR SQL CALL COMPLETED SUCCESSFULLY
****************************************************************
IF SQLCODE NOT = Ø AND SQLCODE NOT = 100 THEN
  MOVE 'BAD' TO RUN-STATUS
  DISPLAY '*    UNEXPECTED SQLCODE FROM SYSPROC.DANACCOR'
      ' DURING ' DB2-COMMAND ' REQUEST.'
  DISPLAY '*'
  PERFORM 9100-DETAIL-SQL-ERROR.
9100-DETAIL-SQL-ERROR.
****************************************************************
* CALL DSNTIAR TO RETURN A TEXT MESSAGE FOR AN UNEXPECTED
* SQLCODE.
****************************************************************
CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
IF RETURN-CODE = ZERO
  PERFORM 9200-PRINT-SQL-ERROR-MSG VARYING ERROR-INDEX
      FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 10.
9200-PRINT-SQL-ERROR-MSG.
****************************************************************
* PRINT MESSAGE TEXT
****************************************************************
DISPLAY ERROR-TEXT (ERROR-INDEX).

COPYIND PROGRAM

IDENTIFICATION DIVISION.
PROGRAM-ID. COPYIND.
AUTHOR. LIJUN GAO;
DATE-WRITTEN. 08/08/03.
DATE-COMPILED.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
/******************************************************************************
DATA DIVISION.
FILE SECTION.
/******************************************************************************
WORKING-STORAGE SECTION.
*******************************************************************************
* DISPLAY FIELD FOR INPUT CRITERIA
*******************************************************************************
Ø1 DIS-CRUPDPGSPCT PIC ZZZZ9.
Ø1 DIS-CRCPYCHGPCT PIC ZZZZ9.
Ø1 DIS-CRDAYSCELSTCPY PIC ZZZZ9.
*******************************************************************************
* OUTPUT TITLE FOR OBJECTS EXCEED COPY CRITERIA LIMITS
*******************************************************************************
Ø1 LIST-COPY-NAMES.
  Ø2 FILLER PIC X(9) VALUE 'DBNAME'.
  Ø2 FILLER PIC X(9) VALUE 'NAME'.
  Ø2 FILLER PIC X(3) VALUE 'TP'.
  Ø2 FILLER PIC X(6) VALUE 'UPG'.
  Ø2 FILLER PIC X(6) VALUE 'CCG'.
  Ø2 FILLER PIC X(6) VALUE 'DSL'.
  Ø2 FILLER PIC X(20) VALUE 'COPY-LASTTIME'.
  Ø2 FILLER PIC X(5) VALUE 'PART'.
*******************************************************************************
* OUTPUT LIST FOR OBJECTS EXCEED COPY LIMITS
*******************************************************************************
Ø1 LIST-COPY.
  Ø2 LIST-COPY-DEF OCCURS 12000 TIMES.
   Ø8 EX-DBNAME PIC X(8).
   Ø8 FILLER PIC X(1).
   Ø8 EX-NAME PIC X(8).
   Ø8 FILLER PIC X(1).
   Ø8 EX-OBJECTTYPE PIC X(2).
   Ø8 FILLER PIC X(1).
   Ø8 EX-CRUPDPGSPCT PIC 9(5).
   Ø8 EX-CRUPG-II REDEFINES EX-CRUPDPGSPCT PIC X(5).
   Ø8 FILLER PIC X(1).
   Ø8 EX-CRCPYCHGPCT PIC 9(5).
   Ø8 EX-CRCG-II REDEFINES EX-CRCPYCHGPCT PIC X(5).
   Ø8 FILLER PIC X(1).
   Ø8 EX-CRDAYSCELSTCPY PIC 9(5).
   Ø8 EX-CRDSL-II REDEFINES EX-CRDAYSCELSTCPY PIC X(5).
   Ø8 FILLER PIC X(1).
   Ø8 EX-COPYLASTTIME PIC X(19).
   Ø8 FILLER PIC X(1).
   Ø8 EX-PARTITION PIC 9(3).
   Ø8 EX-PAR-II REDEFINES EX-PARTITION PIC X(3).
EXEC SQL
  INCLUDE SQLCA
END-EXEC.
EXEC SQL
  INCLUDE WSACCOR
END-EXEC.

PROCEDURE DIVISION USING COPY-REC.
  0000-MAIN-LOGIC.
  PERFORM 1000-INIT THRU 1000-EXIT.
  PERFORM 2100-PROCESS-PARMS THRU 2100-EXIT.
  PERFORM 2200-PROCESS-PARMS THRU 2200-EXIT.
  PERFORM 3000-CONNECT-TO-SERVER THRU 3000-EXIT.
  IF OKAY THEN
    PERFORM 4000-CALL-DSNACCOR THRU 4000-EXIT
  ELSE
    DISPLAY 'CONNECT NOT SUCCESSFUL'
    MOVE 8 TO RETURN-CODE.
  EXEC SQL
CONNECT RESET
END-EXEC.
STOP RUN.
1000-INIT.
MOVE 'GOOD' TO RUN-STATUS.
ACCEPT REFMOD-TIME-ITEM FROM TIME.
ACCEPT YYYYMMDD FROM DATE.
DISPLAY "'DSNG0011 Job execution starting at "
  YYYYMMDD (5:2)
  "/"
  YYYYMMDD (7:2)
  "/2"
  YYYYMMDD (2:3)
  " "
  REFMOD-TIME-ITEM (1:2)
  ":"
  REFMOD-TIME-ITEM (3:2)
  ":"
  REFMOD-TIME-ITEM (5:2)
  "...
DISPLAY 'DSNG0021 MVS=SP7.0.3,PID=HBB7706,DFSMS=1.3.0'
  ',DB2=7.1.0'.
DISPLAY 'DSNG0181 Connected to subsystem ' DB2-LOC-NAME.
1000-EXIT.
EXIT.
2100-PROCESS-PARMS.
* DISPLAY "THE INPUT PARM IS " COPY-REC.
EVALUATE COPY-OBJECT
  WHEN "TYPE"
    MOVE 'TS' TO OBJECTTYPE-DTA
    MOVE 3 TO OBJECTTYPE-LN
  WHEN OTHER
    DISPLAY "'DSNG013E Invalid keyword " COPY-OBJECT
    STOP RUN
END-EVALUATE
EVALUATE COPY-CRI
  WHEN "COPYCRI"
    MOVE 'COPY' TO QUERYTYPE-DTA
    MOVE 8 TO QUERYTYPE-LN
  WHEN OTHER
    DISPLAY "'DSNG013E Invalid keyword " COPY-CRI
    STOP RUN
END-EVALUATE
MOVE COPY-DBNAME-VALUE TO CRI-VALUE.
STRING
  CRI-NAME SPACE CRI-POINT
  DELIMITED BY SIZE
  CRI-VALUE
  DELIMITED BY SPACES
  CRI-POINT
**DELIMITED BY SIZE**

**CRI-EXC**

**DELIMITED BY SIZE**

**INTO CRITERIA-DTA.**

**MOVE 50 TO CRITERIA-LN.**

**IF COPY-CRI-VAL1 NOT EQUAL TO SPACE AND ZERO THEN**

**MOVE COPY-CRI-VAL1 TO CRUPDATEDPAGESPCT.**

**IF COPY-CRI-VAL2 NOT EQUAL TO SPACE AND ZERO THEN**

**MOVE COPY-CRI-VAL2 TO CRCHANGESPCT.**

**IF COPY-CRI-VAL3 NOT EQUAL TO SPACE AND ZERO THEN**

**MOVE COPY-CRI-VAL3 TO CRDAYSNCLASTCOPY.**

**2100-EXIT.**

**EXIT.**

**2200-PROCESS-PARMS.**

****************************

*** PROCESS DSNACCOR INVOCATION PARAMETERS***

****************************

**MOVE 59 TO CHKLVL.**

**DISPLAY " .DSNG015I QueryType = " COPY-TYPE**

**DISPLAY " .DSNG015I ObjectType = " COPY-OBJECT-TYPE**

**IF COPY-OBJECT-TYPE NOT EQUAL TO "TS"**

**DISPLAY ' .DSNG016I Query type COPY will be'**

**' limited to tablespace only'**

**END-IF**

**MOVE CRUPDATEDPAGESPCT TO DIS-CRUPDPGPSCT.**

**MOVE CRCHANGESPCT TO DIS-CRCPYCHGPCT**

**MOVE CRDAYSNCLASTCOPY TO DIS-CRDAYSCELSTCPY**

**DISPLAY " .DSNG015I CRUDPGPSCT = " DIS-CRUPDPGPSCT**

**DISPLAY " .DSNG015I CRCPYCHGPCT = " DIS-CRCPYCHGPCT**

**DISPLAY " .DSNG015I CRDAYSCELSTCPY = " DIS-CRDAYSCELSTCPY**

**DISPLAY " .DSNG015I QUERYSCOPE = WHERE " CRITERIA-DTA.**

**DISPLAY ' .'.**

*******************************

*** INITIALIZE OUTPUT PARAMETERS * ***

*******************************

**MOVE SPACES TO LASTSTATEMENT-DTA.**

**MOVE 1 TO LASTSTATEMENT-LN.**

**MOVE Ø TO RETURNCODE.**

**MOVE SPACES TO ERRORMSG-DTA.**

**MOVE 1 TO ERRORMSG-LN.**

**MOVE Ø TO IFCARETCODE.**

**MOVE Ø TO IFCARESCODE.**

**MOVE Ø TO XSBYTES.**

*******************************

*** SET THE INDICATOR VARIABLES TO Ø FOR NON-NULL INPUT * ***

**PARAMETERS (PARAMETERS FOR WHICH YOU DO NOT WANT **

**DSNACCOR TO USE DEFAULT VALUES) AND FOR OUTPUT **

**PARAMETERS.**

*******************************

**MOVE Ø TO CHKLVL-IND.**
MOVE Ø TO CRITERIA-IND.
MOVE Ø TO CRUPDATERESPCT-IND.
MOVE Ø TO CRCHEANGESPCT-IND.
MOVE Ø TO CRDAYSNCLASTCOPY-IND.
MOVE Ø TO LASTSTATEMENT-IND.
MOVE Ø TO RETURNCODE-IND.
MOVE Ø TO ERORRMSG-IND.
MOVE Ø TO IFCARETCODE-IND.
MOVE Ø TO IFCARESCODE-IND.
MOVE Ø TO XSBYTES-IND.
2200-EXIT.
EXIT.
3000-CONNECT-TO-SERVER.
****************************************************************
* CONNECT TO THE REMOTE SERVER
****************************************************************
EXEC SQL CONNECT TO :DB2-LOC-NAME END-EXEC.
MOVE 'CONNECT' TO DB2-COMMAND.
IF SQLCODE IS NOT EQUAL TO ZERO THEN
   PERFORM 9000-CHECK-SQLCODE.
3000-EXIT.
EXIT.
4000-CALL-DSNACCOR.
****************************************************************
* CALL DSNACCOR *
****************************************************************
EXEC SQL CALL DSNACCOR
(:QUERYTYPE :QUERYTYPE-IND,
 :OBJECTTYPE :OBJECTTYPE-IND,
 :ICTYPE :ICTYPE-IND,
 :STATSSCHEMA :STATSSCHEMA-IND,
 :CATLGSHEMA :CATLGSHEMA-IND,
 :LOCALSCHEMA :LOCALSCHEMA-IND,
 :CHKLVL :CHKLVL-IND,
 :CRITERIA :CRITERIA-IND,
 :RESTRICTED :RESTRICTED-IND,
 :CRUPDATERESPCT :CRUPDATERESPCT-IND,
 :CRCHEANGESPCT :CRCHEANGESPCT-IND,
 :CRDAYSNCLASTCOPY :CRDAYSNCLASTCOPY-IND,
 :ICRUPDATERESPCT :ICRUPDATERESPCT-IND,
 :ICRCHEANGESPCT :ICRCHEANGESPCT-IND,
 :CRINDEXSIZE :CRINDEXSIZE-IND,
 :RRTINSDELUPDPCT :RRTINSDELUPDPCT-IND,
 :RRTUNCLUSTINSPECT :RRTUNCLUSTINSPECT-IND,
 :RRTDISORGLOBPCT :RRTDISORGLOBPCT-IND,
 :RRTMASSDELLIMIT :RRTMASSDELLIMIT-IND,
 :RRTINDREFLIMIT :RRTINDREFLIMIT-IND,
 :RRIINSERTDELETEPCT :RRIINSERTDELETEPCT-IND,
 :RIAPPENDINSERTPCT :RIAPPENDINSERTPCT-IND,
 :RRIPSEUDODELETEPCT :RRIPSEUDODELETEPCT-IND,
MOVE 'CALL' TO DB2-COMMAND.
IF SQLCODE IS NOT EQUAL TO +466 THEN
   PERFORM 9ØØØ-CHECK-SQLCODE
ELSE
   PERFORM 41ØØ-GET-RESULT.
4ØØØ-EXIT.
EXIT.
41ØØ-GET-RESULT.
IF RETURNCODE NOT EQUAL TO Ø THEN
   DISPLAY 'RETURNCODE' RETURNCODE
   DISPLAY 'ERRORMSG' ERRORMSG
   DISPLAY 'LASTSTATEMENT' LASTSTATEMENT
   DISPLAY IFCARETCODE IFCARESCODE XSBYTES
ELSE
   DISPLAY '.DSNGØ1I' ERRORMSG.
EXEC SQL ASSOCIATE LOCATORS(:LOC1, :LOC2)
   WITH PROCEDURE DSNACCOR
END-EXEC.
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :LOC1
END-EXEC
EXEC SQL ALLOCATE C2 CURSOR FOR RESULT SET :LOC2
END-EXEC
PERFORM 4Ø5Ø-DIS-TITLE
PERFORM 43ØØ-GET-RESULT2 THRU 43ØØ-EXIT VARYING I FROM 1 BY 1 UNTIL SQLCODE EQUAL TO +1ØØ.
DISPLAY '------------------------------------------------'
'.------------------------------'.
COMPUTE I = I - 1.
COMPUTE J = J - 1.
MOVE I TO DIS-I.
MOVE J TO DIS-J.
DISPLAY '.DSNGØ2I ' TOTAL ' DIS-I ' RECORDS RETRIEVED AND ' DIS-J ' RECORDS DISPLAYED.'.
**4Ø5Ø-DIS-TITLE.**

DISPLAY '**************************************************'
    '*****************************'
DISPLAY '*  UPG: The ratio of distinct updated pages to preformatted pages*
DISPLAY '*  CCG: the number of INS, UPD and DEL to the total number of rows *
DISPLAY '*  DSL: the number of days since the last image copy *
DISPLAY '**************************************************'
    '*****************************'

DISPLAY LIST-COPY-NAMES.
DISPLAY '-----------------------------------------------'
    '------------------------------'.

**43ØØ-GET-RESULT2.**

EXEC SQL FETCH C2 INTO :RS-OUTPUT-2 :RS-OUT2-IND
END-EXEC.
MOVE 'FETCH' TO DB2-COMMAND.
PERFORM 90ØØ-CHECK- SQLCODE.
MOVE RS-DBNAME TO EX-DBNAME(I).
MOVE RS-NAME TO EX-NAME(I).
MOVE RS-OBJECTTYPE TO EX-OBJECTTYPE(I).
MOVE RS-ASSOCIATEDTS TO EX-ASSOCIATEDTS(I).
IF RS-OUT2-IND(12) NOT < Ø THEN
    MOVE RS-COPYLASTTIME TO EX-COPYLASTTIME(I)
ELSE
    GO TO 43ØØ-EXIT.
IF RS-OUT2-IND(15) < Ø OR RS-CRUPDPGSPCT EQUAL TO Ø THEN
    MOVE LOW-VALUES TO EX-CRUPG-II(I)
ELSE
    MOVE RS-CRUPDPGSPCT TO EX-CRUPDPGSPCT(I).
IF RS-RCRPYCHGPCT EQUAL TO Ø OR RS-OUT2-IND(16) < Ø THEN
    MOVE LOW-VALUES TO EX-CRCCG-II(I)
ELSE
    MOVE RS-RCRPYCHGPCT TO EX-CRCPYCHGPCT(I).
IF RS-CRDAYSCELSTOPY EQUAL TO Ø OR RS-OUT2-IND(17) < Ø THEN
    MOVE LOW-VALUES TO EX-CRDSL-II(I)
ELSE
    MOVE RS-CRDAYSCELSTOPY TO EX-CRDAYSCESLSTOPY(I).
IF RS-PARTITION EQUAL TO Ø THEN
    MOVE LOW-VALUES TO EX-PAR-II(I)
ELSE
    MOVE RS-PARTITION TO EX-PARTITION(I).
DISPLAY LIST-COPY-DEF(I).
COMPUTE J = J + 1.

**43ØØ-EXIT.**
EXIT.

**90ØØ-CHECK-SQLCODE.**

*******************************************************************************
* VERIFY THAT THE PRIOR SQL CALL COMPLETED SUCCESSFULLY
****************************************************************
IF SQLCODE NOT = 0 AND SQLCODE NOT = 100 THEN
  MOVE 'BAD' TO RUN-STATUS
  DISPLAY '*    UNEXPECTED SQLCODE FROM SYSPROC.DANACCOR'
    ' DURING ' DB2-COMMAND ' REQUEST.'
  DISPLAY '*'
  PERFORM 9100-DETAIL-SQL-ERROR.
****************************************************************
* CALL DSNTIAR TO RETURN A TEXT MESSAGE FOR AN UNEXPECTED
* SQLCODE.
****************************************************************
CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
  IF RETURN-CODE = ZERO
    PERFORM 9200-PRINT-SQL-ERROR-MSG VARYING ERROR-INDEX
      FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 10.
  9200-PRINT-SQL-ERROR-MSG.
****************************************************************
* PRINT MESSAGE TEXT
****************************************************************
DISPLAY ERROR-TEXT (ERROR-INDEX).

RUNSTIND COBOL

IDENTIFICATION DIVISION.
PROGRAM-ID. RUNSTIND.
AUTHOR. LIJUN GAO;
DATE-WRITTEN. 08/08/03.
DATE-COMPILED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
DISPLAY FIELDS FOR INPUT CRITERIA
****************************************************************
01 DIS-SRTNSDELUPDPCT PIC ZZZ9.
01 DIS-SRTNSDELUDPABS PIC ZZZ9.
01 DIS-SRTMASSDELLIMIT PIC ZZZ9.
01 DIS-SRIINSDELUPDPCT PIC ZZZ9.
Ø1 DIS-SRIINSDELUPDABS  PIC ZZZ9.
Ø1 DIS-SRIMASSDELLIMIT  PIC ZZZ9.
*****************************************************************************
* OUTPUT TITLE FOR OBJECTS EXCEED RUNSTA CRITERIA LIMITS
*****************************************************************************
Ø1 LIST-RUNSTA-NAMES.
Ø2 FILLER               PIC X(9) VALUE 'DBNAME'.
Ø2 FILLER               PIC X(9) VALUE 'NAME'.
Ø2 FILLER               PIC X(3) VALUE 'TP'.
Ø2 FILLER               PIC X(6) VALUE 'SRIDU'.
Ø2 FILLER               PIC X(6) VALUE 'SRIDA'.
Ø2 FILLER               PIC X(6) VALUE 'SRMDL'.
Ø2 FILLER               PIC X(20) VALUE 'RUNSTAT-LASTTIME'.
Ø2 FILLER               PIC X(5) VALUE 'PART'.
Ø2 FILLER               PIC X(10) VALUE 'ASSOC-TS'.
*****************************************************************************
* OUTPUT LIST FOR OBJECTS EXCEED RUNSTA LIMITS
*****************************************************************************
Ø1 LIST-RUNSTA.
Ø2 LIST-RUNSTA-DEF OCCURS 12000 TIMES.
 Ø8 EX-DBNAME            PIC X(8).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-NAME              PIC X(8).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-OBJECTTYPE        PIC X(2).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-SRTINSDELPCT      PIC 9(5).
 Ø8 EX-SRIDU-II REDEFINES EX-SRTINSDELPCT PIC X(5).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-SRTINSDELABS      PIC 9(5).
 Ø8 EX-SRIDA-II REDEFINES EX-SRTINSDELABS PIC X(5).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-SRMASSDELETE      PIC 9(5).
 Ø8 EX-SRMDL-II REDEFINES EX-SRMASSDELETE PIC X(5).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-RUNSTALASTTIME    PIC X(19).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-PARTITION         PIC 9(3).
 Ø8 EX-PAR-II REDEFINES EX-PARTITION PIC X(3).
 Ø8 FILLER               PIC X(1).
 Ø8 EX-ASSOCIATEDTS      PIC X(8).
 Ø8 FILLER               PIC X(1).
*****************************************************************************
* COPY ALL RELATED WORKING STORAGE DEFINITION
*****************************************************************************
COPY WRKINPT.
*****************************************************************************
* DB2 AREA
*****************************************************************************
EXEC SQL
INCLUDE SQLCA
END-EXEC.
EXEC SQL
  INCLUDE WSACCOR
END-EXEC.
LINKAGE SECTION.
  01 RUNSTA-REC.
  05 LINEA.
    07 RUNSTA-TYPE PIC X(08).
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-DBNAME PIC X(08).
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-DBNAME-VALUE PIC X(08).
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-OBJECT PIC X(4).
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-OBJECT-TYPE PIC X(3).
    07 FILLER PIC X VALUE SPACE.
  05 LINEB.
    07 RUNSTA-CRI PIC X(08).
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL1 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL2 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL3 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL4 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL5 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.
    07 RUNSTA-CRI-VAL6 PIC 9(4) VALUE ZERO.
    07 FILLER PIC X VALUE SPACE.

PROCEDURE DIVISION USING RUNSTA-REC.
  0000-MAIN-LOGIC.
    PERFORM 1000-INIT THRU 1000-EXIT.
    PERFORM 2100-PROCESS-PARMS THRU 2100-EXIT.
    PERFORM 2200-PROCESS-PARMS THRU 2200-EXIT.
    PERFORM 3000-CONNECT-TO-SERVER THRU 3000-EXIT.
    IF OKAY THEN
      PERFORM 4000-CALL-DSNACCOR THRU 4000-EXIT
    ELSE
      DISPLAY 'CONNECT NOT SUCCESSFUL'.
      MOVE 8 TO RETURN-CODE.
    END-EXEC.
    STOP RUN.
  1000-INIT.
    MOVE 'GOOD' TO RUN-STATUS.
ACCEPT REFMOD-TIME-ITEM FROM TIME.
ACCEPT YYYYMMDD FROM DATE.
DISPLAY "`.DSNG001I Job execution starting at "
  YYYYMMDD (5:2)
    "/"
  YYYYMMDD (7:2)
    "/"2"
  YYYYMMDD (2:3)
    " "
  REFMOD-TIME-ITEM (1:2)
    ":"
  REFMOD-TIME-ITEM (3:2)
    ":"
  REFMOD-TIME-ITEM (5:2)
    " ..."
DISPLAY ``.DSNG002I MVS=SP7.0.3,PID=HBB7706,DFSMS=1.3.0'
    ',DB2=7.1.0'.
DISPLAY ``.DSNG018I Connected to subsystem 'DB2-LOC-NAME.'
1000-EXIT.
EXIT.
2100-PROCESS-PARMS.
EVALUATE RUNSTA-OBJECT
  WHEN "TYPE"
    MOVE RUNSTA-OBJECT-TYPE TO OBJECTTYPE-DTA
    MOVE 3 TO OBJECTTYPE-LN
  WHEN OTHER
    DISPLAY ``.DSNG013E Invalid keyword " RUNSTA-OBJECT
    STOP RUN
END-EVALUATE
EVALUATE RUNSTA-CRI
  WHEN "RUNCRI"
    MOVE `RUNSTATS' TO QUERYTYPE-DTA
    MOVE 8 TO QUERYTYPE-LN
  WHEN OTHER
    DISPLAY ``.DSNG013E Invalid keyword " RUNSTA-CRI
    STOP RUN
END-EVALUATE
MOVE RUNSTA-DBNAME-VALUE TO CRI-VALUE.
STRING
  CRI-NAME SPACE CRI-POINT
  DELIMITED BY SIZE
  CRI-VALUE
  DELIMITED BY SPACES
  CRI-POINT
  DELIMITED BY SIZE
  CRI-EXC
  DELIMITED BY SIZE
  INTO CRITERIA-DTA.
MOVE 50 TO CRITERIA-LN.
IF RUNSTA-CRI-VAL1 NOT EQUAL TO SPACE AND ZERO THEN

MOVE RUNSTA-CRI-VAL1 TO SRTINSDELUPDPCT.
  IF RUNSTA-CRI-VAL2 NOT EQUAL TO SPACE AND ZERO THEN
    MOVE RUNSTA-CRI-VAL2 TO SRTINSDELUPDABS.
  IF RUNSTA-CRI-VAL3 NOT EQUAL TO SPACE AND ZERO THEN
    MOVE RUNSTA-CRI-VAL3 TO SRTMASSDELLIMIT.
  IF RUNSTA-CRI-VAL4 NOT EQUAL TO SPACE AND ZERO THEN
    MOVE RUNSTA-CRI-VAL4 TO SRIINSDELUPDPCT.
  IF RUNSTA-CRI-VAL5 NOT EQUAL TO SPACE AND ZERO THEN
    MOVE RUNSTA-CRI-VAL5 TO SRIINSDELUPDABS.
  IF RUNSTA-CRI-VAL6 NOT EQUAL TO SPACE AND ZERO THEN
    MOVE RUNSTA-CRI-VAL6 TO SRIMASSDELLIMIT.
2100-EXIT.
  EXIT.
2200-PROCESS-PARMS.
  ****************************************************
* PROCESS DSNACCOR INVOCATION PARAMETERS
  ****************************************************
  MOVE 59 TO CHKLVL.
  DISPLAY "DSNG015I QueryType = " RUNSTA-type
  DISPLAY "DSNG015I ObjectType = " RUNSTA-object-type
  MOVE SRTINSDELUPDPCT TO DIS-SRTINSDELUPDPCT.
  MOVE SRTINSDELUPDABS TO DIS-SRTINSDELUPDABS
  MOVE SRTMASSDELLIMIT TO DIS-SRTMASSDELLIMIT
  MOVE SRIINSDELUPDPCT TO DIS-SRIINSDELUPDPCT.
  MOVE SRIINSDELUPDABS TO DIS-SRIINSDELUPDABS
  MOVE SRIMASSDELLIMIT TO DIS-SRIMASSDELLIMIT
  IF RUNSTA-OBJECT-TYPE = "ALL" OR "TS" THEN
    DISPLAY "DSNG015I SRTINSDELPCT = " DIS-SRTINSDELUPDPCT
    DISPLAY "DSNG015I SRTINSDELABS = " DIS-SRTINSDELUPDABS
    DISPLAY "DSNG015I SRTMASSDELETE = " DIS-SRTMASSDELLIMIT.
  IF RUNSTA-OBJECT-TYPE = "ALL" OR "IX" THEN
    DISPLAY "DSNG015I SRIINSDELPCT = " DIS-SRIINSDELUPDPCT
    DISPLAY "DSNG015I SRIINSDELABS = " DIS-SRIINSDELUPDABS
    DISPLAY "DSNG015I SRIMASSDELETE = " DIS-SRIMASSDELLIMIT.
    DISPLAY "DSNG015I QUERYSCOPE = WHERE " CRITERIA-DTA.
    DISPLAY '.
  ****************************************************
* INITIALIZE OUTPUT PARAMETERS *
  ******************************************************
  MOVE SPACES TO LASTSTATEMENT-DTA.
  MOVE 1 TO LASTSTATEMENT-LN.
  MOVE Ø TO RETURNCODE.
  MOVE SPACES TO ERRORMSG-DTA.
  MOVE 1 TO ERRORMSG-LN.
  MOVE Ø TO IFCARETCODE.
  MOVE Ø TO IFCARESCODE.
  MOVE Ø TO XSBYTES.
  ******************************************************
* SET THE INDICATOR VARIABLES TO Ø FOR NON-NULL INPUT *
* PARAMETERS (PARAMETERS FOR WHICH YOU DO NOT WANT  *
* DSNACCOR TO USE DEFAULT VALUES) AND FOR OUTPUT      *
* PARAMETERS.                                         *
*******************************************************
MOVE Ø TO CHKLVL-IND.
MOVE Ø TO CRITERIA-IND.
MOVE Ø TO SRTINSDELUPDPCT-IND.
MOVE Ø TO SRTINSDELUPDABS-IND.
MOVE Ø TO SRTMASSDELLIMIT-IND.
MOVE Ø TO SRIINSDELUPDPCT-IND.
MOVE Ø TO SRIINSDELUPDABS-IND.
MOVE Ø TO SRIIMASSDELLIMIT-IND.
MOVE Ø TO LASTSTATEMENT-IND.
MOVE Ø TO RETURNCODE-IND.
MOVE Ø TO ERRORMSG-IND.
MOVE Ø TO IFCARETCODE-IND.
MOVE Ø TO IFCARESCODE-IND.
MOVE Ø TO XSBYTES-IND.

2200-EXIT.
EXIT.

3000-CONNECT-TO-SERVER.
****************************************************************
* CONNECT TO THE REMOTE SERVER
****************************************************************
EXEC SQL CONNECT TO :DB2-LOC-NAME END-EXEC.
MOVE 'CONNECT' TO DB2-COMMAND.
IF SQLCODE IS NOT EQUAL TO ZERO THEN
    PERFORM 9000-CHECK-SQLCODE.
3000-EXIT.
EXIT.

4000-CALL-DSNACCOR.
*******************************************************************************
* CALL DSNACCOR *
*******************************************************************************
EXEC SQL CALL DSNACCOR
  (:QUERYTYPE           :QUERYTYPE-IND,
   :OBJECTTYPE          :OBJECTTYPE-IND,
   :ICTYPE              :ICTYPE-IND,
   :STATSSCHEMA         :STATSSCHEMA-IND,
   :CATLGSCHEMA         :CATLGSCHEMA-IND,
   :LOCALSCHEMA         :LOCALSCHEMA-IND,
   :CHKLVL              :CHKLVL-IND,
   :CRITERIA            :CRITERIA-IND,
   :RESTRICTED          :RESTRICTED-IND,
   :CRUPDATEDPAGESPCT   :CRUPDATEDPAGESPCT-IND,
   :CRCHANGESPCT        :CRCHANGESPCT-IND,
   :CRDAYSNCLASTCOPY    :CRDAYSNCLASTCOPY-IND,
   :ICRUPDATEDPAGESPCT  :ICRUPDATEDPAGESPCT-IND,
   :ICRCHANGESPCT       :ICRCHANGESPCT-IND,
   :CRINDEXSIZE         :CRINDEXSIZE-IND,
   :RRTINSDELUPDPCT     :RRTINSDELUPDPCT-IND,
MOVE 'CALL' TO DB2-COMMAND.
IF SQLCODE IS NOT EQUAL TO +466 THEN
   PERFORM 9ØØØ-CHECK-SQLCODE
ELSE
   PERFORM 41ØØ-GET-RESULT.

4ØØØ-EXIT.
EXIT.

41ØØ-GET-RESULT.
IF RETURNCODE NOT EQUAL TO Ø THEN
   DISPLAY 'LASTSTATEMENT' LASTSTATEMENT
   DISPLAY 'ERRORMSG' ERRORMSG
   DISPLAY 'RETURNCODE' RETURNCODE
   DISPLAY IFCARETCODE IFCARESCODE XSBYTES
ELSE
   DISPLAY '.DSNGØ11I ' ERRORMSG.
EXEC SQL ASSOCIATE LOCATORS(:LOC1,  :LOC2)
   WITH PROCEDURE DSNACCOR
END-EXEC.
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :LOC1
END-EXEC
EXEC SQL ALLOCATE C2 CURSOR FOR RESULT SET :LOC2
END-EXEC
PERFORM 4Ø5Ø-DIS-TITLE
PERFORM 43ØØ-GET-RESULT2 THRU 43ØØ-EXIT VARYING I FROM 1 BY 1 UNTIL SQLCODE EQUAL TO +1Ø0.
DISPLAY '------------------------------------------------'
MOVE I TO DIS-I.
MOVE J TO DIS-J.
DISPLAY '.DSNGØ2I1I TOTAL ' DIS-I ' RECORDS RETRIEVED AND ' DIS-J ' RECORDS DISPLAYED.'.

4Ø5Ø-DIS-TITLE.
DISPLAY '*****************************************************
***********************
DISPLAY '* SRIDU: The ratio fo the number of INS, UPD'
', DEL to total number of rows   *'
DISPLAY '* SRIDA: The number INS, UPD and DEL since t'
'he last RUNSTATS  *'
DISPLAY '* SRMDL: The number of mass deletes since la'
'st REORG or LOAD REPLACE  *'
DISPLAY '*****************************************************
***********************
DISPLAY '.
DISPLAY LIST-RUNSTA-NAMES.
DISPLAY '------------------------------------------------
------------------------------'.

43ØØ-GET-RESULT2.
EXEC SQL FETCH C2 INTO :RS-OUTPUT-2 :RS-OUT2-IND
END-EXEC.
MOVE 'FETCH' TO DB2-COMMAND.
PERFORM 90ØØ-CHECK-SQLCODE.
IF RS-OUT2-IND(31) NOT < Ø THEN
   MOVE RS-STATSLASTTIME TO EX-RUNSTALASTTIME(I)
ELSE
   GO TO 43ØØ-EXIT.
END-IF
MOVE RS-DBNAME TO EX-DBNAME(I).
MOVE RS-NAME TO EX-NAME(I).
MOVE RS-OBJECTTYPE TO EX-OBJECTTYPE(I).
MOVE RS-ASSOCIATEDTS TO EX-ASSOCIATEDTS(I).
IF RS-PARTITION EQUAL TO Ø THEN
   MOVE LOW-VALUES TO EX-PAR-II(I)
ELSE
   MOVE RS-PARTITION TO EX-PARTITION(I).
END-IF
EVALUATE RS-OBJECTTYPE
WHEN "TS"
   IF RS-OUT2-IND(32) < Ø OR RS-SRTINSDELPCT = Ø THEN
      MOVE LOW-VALUES TO EX-SRIDU-II(I)
   ELSE
      MOVE RS-SRTINSDELPCT TO EX-SRTINSDELPCT(I)
   END-IF
   IF RS-OUT2-IND(33) < Ø OR RS-SRTINSDELABS = Ø THEN
      MOVE LOW-VALUES TO EX-SRIDA-II(I)
   ELSE
      MOVE RS-SRTINSDELABS TO EX-SRTINSDELABS(I)
   END-IF
   IF RS-OUT2-IND(34) < Ø OR RS-SRTMASSDELETE = Ø THEN

MOVE LOW-VALUES TO EX-SRMDL-II(I)
ELSE
    MOVE RS-SRTMASSDELETE TO EX-SRTMASSDELETE(I)
END-IF
WHEN "IX"
IF RS-OUT2-IND(35) < Ø OR RS-SRIINSDELPCT EQUAL TO Ø THEN
    MOVE LOW-VALUES TO EX-SRIDU-II(I)
ELSE
    MOVE RS-SRIINSDELPCT TO EX-SRTINSDELPCT(I)
END-IF
IF RS-OUT2-IND(36) < Ø OR RS-SRIINSDELABS = Ø THEN
    MOVE LOW-VALUES TO EX-SRIDA-II(I)
ELSE
    MOVE RS-SRIINSDELABS TO EX-SRTINSDELABS(I)
END-IF
IF RS-OUT2-IND(37) < Ø OR RS-SRIMASSDELETE = Ø THEN
    MOVE LOW-VALUES TO EX-SRMDL-II(I)
ELSE
    MOVE RS-SRIMASSDELETE TO EX-SRTMASSDELETE(I)
END-IF
WHEN OTHER
    CONTINUE
END-EVALUATE
DISPLAY LIST-RUNSTA-DEF(I).
COMPUTE J = J + 1.
4300-EXIT.
EXIT.
9000-CHECK-SQLCODE.
****************************************************************
* VERIFY THAT THE PRIOR SQL CALL COMPLETED SUCCESSFULLY
****************************************************************
IF SQLCODE NOT = Ø AND SQLCODE NOT = 100 THEN
    MOVE 'BAD' TO RUN-STATUS
    DISPLAY '*    UNEXPECTED SQLCODE FROM SYSPROC.DANACCOR' ' DURING ' DB2-COMMAND ' REQUEST.'
    DISPLAY '*'
    PERFORM 9100-DETAIL-SQL-ERROR.
9100-DETAIL-SQL-ERROR.
****************************************************************
* CALL DSNTIAR TO RETURN A TEXT MESSAGE FOR AN UNEXPECTED SQLCODE.
****************************************************************
CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
IF RETURN-CODE = ZERO
    PERFORM 9200-PRINT-SQL-ERROR-MSG VARYING ERROR-INDEX FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 10.
9200-PRINT-SQL-ERROR-MSG.
****************************************************************
* PRINT MESSAGE TEXT
****************************************************************
    DISPLAY ERROR-TEXT (ERROR-INDEX).
WRKINPT COBOL

Ø1 WS-IDX                         PIC 9(3) VALUE 1.
Ø1 WS-IDX2                        PIC 9(3) VALUE 1.
Ø1 WS-IDX-MAX                     PIC 9(3) VALUE 1.
Ø1 YYYYMMDD                       PIC 9(8).
Ø1 REFMOD-TIME-ITEM               PIC X(8).
Ø1 WRK-CRITERIA.
   Ø5 CRI-NAME           PIC X(12) VALUE "DBNAME LIKE ".
   Ø5 CRI-VALUE          PIC X(8).
   Ø5 CRI-POINT          PIC X(1) VALUE "'".
   Ø5 CRI-EXC PIC X(27) VALUE " AND DBNAME NOT LIKE 'DSN%'".
77 I                        PIC 9(6)   COMP.
77 J                        PIC 9(6)   COMP VALUE 1.
77 DIS-I                    PIC ZZZZZ9.
77 DIS-J                    PIC ZZZZZ9.

*****************************************************************
* JOB STATUS INDICATOR
*****************************************************************
Ø1 RUN-STATUS               PIC X(4).
   88 NOT-OKAY                          VALUE 'BAD'.
   88 OKAY                              VALUE 'GOOD'.
*****************************************************************
* BUFFER FOR RECEIVING SQL ERROR MESSAGES
*****************************************************************
Ø1 ERROR-MESSAGE.
   Ø2  ERROR-LEN            PIC S9(4)   COMP VALUE +960.
   Ø2  ERROR-TEXT           PIC X(120)  OCCURS 10 TIMES
                                 INDEXED BY ERROR-INDEX.
77 ERROR-TEXT-LEN           PIC S9(9)   COMP VALUE +120.

WSACCOR CODE

*******************************
* DSNACCOR PARAMETERS *
*******************************
Ø1 QUERYTYPE.
   49 QUERYTYPE-LN       PICTURE S9(4) COMP VALUE 40.
   49 QUERYTYPE-DTA      PICTURE X(40)  VALUE 'ALL'.
Ø1 OBJECTTYPE.
   49 OBJECTTYPE-LN      PICTURE S9(4) COMP VALUE 3.
   49 OBJECTTYPE-DTA     PICTURE X(3)  VALUE 'ALL'.
Ø1 ICTYPE.
   49 ICTYPE-LN          PICTURE S9(4) COMP VALUE 1.
   49 ICTYPE-DTA         PICTURE X(1)  VALUE 'F'.
Ø1 STATSSCHEMA.
   49 STATSSCHEMA-LN     PICTURE S9(4) COMP VALUE 128.
   49 STATSSCHEMA-DTA    PICTURE X(128) VALUE 'SYSIBM'.
Ø1 CATLGSHEMA.
49 CATLGSHEMA-LN PICTURE S9(4) COMP VALUE 128.
49 CATLGSHEMA-DTA PICTURE X(128) VALUE 'SYSIBM'.
01 LOCALSCHEMA.
  49 LOCALSCHEMA-LN PICTURE S9(4) COMP VALUE 128.
  49 LOCALSCHEMA-DTA PICTURE X(128) VALUE 'DSNACC'.
01 CHKLV.
  49 CRITERIA-LN PICTURE S9(4) COMP VALUE 80.
  49 CRITERIA-DTA PICTURE X(80) VALUE SPACES.
01 RESTRICTED.
  49 RESTRICTED-LN PICTURE S9(4) COMP VALUE 80.
  49 RESTRICTED-DTA PICTURE X(80) VALUE SPACES.
01 CRUPDATEDPAGESPCT PICTURE S9(9) COMP VALUE +20.
01 CRCHANGESPCT PICTURE S9(9) COMP VALUE +10.
01 CRDATESNCLASTCOPY PICTURE S9(9) COMP VALUE +7.
01 ICRUPDATEDPAGESPCT PICTURE S9(9) COMP VALUE +0.
01 IRCRCHANGE SPCT PICTURE S9(9) COMP VALUE +0.
01 CRINDEXSIZE PICTURE S9(9) COMP VALUE +0.
01 RRINSDELELPDCT PICTURE S9(5) COMP VALUE +20.
01 RRTUNCLUDINSPCT PICTURE S9(5) COMP VALUE +10.
01 RTDISOR GLOB PCT PICTURE S9(5) COMP VALUE +10.
01 RRTMASSDELLIMIT PICTURE S9(5) COMP VALUE +0.
01 RRindexes LIMIT PICTURE S9(5) COMP VALUE +10.
01 RIIINSERTDELET PCT PICTURE S9(9) COMP VALUE +20.
01 RRAPPENDIN SERT PCT PICTURE S9(9) COMP VALUE +10.
01 RRIINDEX PSEUDO PCT PICTURE S9(9) COMP VALUE +10.
01 RRIMA SDELLIMIT PICTURE S9(9) COMP VALUE +0.
01 RRINDEX LIMIT PICTURE S9(9) COMP VALUE +10.
01 RRINUMLEVELSLIMIT PICTURE S9(9) COMP VALUE +0.
01 SRINSDELUP DCT PICTURE S9(9) COMP VALUE +20.
01 SRIDELUPDABS PICTURE S9(9) COMP VALUE +0.
01 SRTMASSDELLIMIT PICTURE S9(9) COMP VALUE +0.
01 SRIINSDELP ASCHT PICTURE S9(9) COMP VALUE +20.
01 SRIINSDELUPSABS PICTURE S9(9) COMP VALUE +0.
01 SRIINSDELUPDABS PICTURE S9(9) COMP VALUE +0.
01 SRIASSDELLIMIT PICTURE S9(9) COMP VALUE +0.
01 EXTENT LIMIT PICTURE S9(4) COMP VALUE +50.
01 LASTSTATEMENT.
  49 LASTSTATEMENT-LN PICTURE S9(4) COMP VALUE 8012.
  49 LASTSTATEMENT-DTA PICTURE X(8012) VALUE SPACES.
01 RETURNCODE PICTURE S9(9) COMP VALUE +0.
01 ERRORMSG.
  49 ERRORMSG-LN PICTURE S9(4) COMP VALUE 240.
  49 ERRORMSG-DTA PICTURE X(240) VALUE SPACES.
01 IFCARETCODE PICTURE S9(9) COMP VALUE +0.
01 IFCARESCODE PICTURE S9(9) COMP VALUE +0.
01 XSBYTES PICTURE S9(9) COMP VALUE +0.

********************************************************************************
* INDICATOR VARIABLES.          *
* INITIALIZE ALL NON-ESSENTIAL INPUT  *
* VARIABLES TO -1, TO INDICATE THAT THE *
* INPUT VALUE IS NULL.  *
********************************************************************
Ø1  QUERYTYPE-IND    PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  OBJECTTYPE-IND   PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  ICTYPE-IND       PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  STATSSCHEMA-IND  PICTURE S9(4) COMP-4 VALUE -1.
Ø1  CATLGSHEMA-IND   PICTURE S9(4) COMP-4 VALUE -1.
Ø1  LOCALSCHEMA-IND  PICTURE S9(4) COMP-4 VALUE -1.
Ø1  CRUPDATEDPAGESPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  CRCHANGESPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  CRINDEXSIZE-IND  PICTURE S9(4) COMP-4 VALUE -1.
Ø1  RRTINSDELUPDPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  RRTINSDELETEPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  SRTINSDELUPDPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  SRTINSDELETEPCT-IND PICTURE S9(4) COMP-4 VALUE -1.
Ø1  LASTSTATEMENT-IND PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  RETURNCODE-IND   PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  ERRORMSG-IND     PICTURE S9(4) COMP-4 VALUE +Ø.
Ø1  XSBYTES-IND      PICTURE S9(4) COMP-4 VALUE +Ø.
********************************************************************
* OUTPUT RESULT SET
********************************************************************
Ø1  RS-OUTPUT-1.
  Ø2  RS-SEQUENCE   PIC S9(9) COMP-4 VALUE +Ø.
  Ø2  RS-DATA       PIC X(ØØ).
Ø1  RS-OUTPUT-2.
  Ø2  RS-DBNAME     PIC X(8).
  Ø2  RS-NAME       PIC X(8).

Ø2 RS-PARTITION    PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-OBJECTTYPE   PIC X(2).
Ø2 RS-OBJECTSTATUS PIC X(36).
Ø2 RS-IMAGECOPY    PIC X(3).
Ø2 RS-RUNSTATS    PIC X(3).
Ø2 RS-EXTENTS     PIC X(3).
Ø2 RS-REORG       PIC X(3).
Ø2 RS-INEXCEPTTABLE PIC X(4Ø).
Ø2 RS-ASSOCIATEDTS PIC X(Ø).
Ø2 RS-COPYLASTTIME PIC X(26).
Ø2 RS-LOADRLASTTIME PIC X(26).
Ø2 RS-REBUILDLASTTIME PIC X(26).
Ø2 RS-CRUPDPGSPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-CRCPYCHGPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-CRDAYSCELSTCPY PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-CRINDEXSIZE PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-REORGLASTTIME PIC X(26).
Ø2 RS-RRTINSDELPDPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRTUNCINSPECT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRTDISORGLGBPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRTMASSDELETE PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRTINDREF PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRIINSDELPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRIAPPINSPECT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRIPSDELPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRIMASSDELETE PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRILEAF PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-RRINUMLEVELS PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-STATSLASTTIME PIC X(26).
Ø2 RS-SRTINSDELPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-SRTINSDELABS PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-SRTMASSDELETE PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-SRIINSDELPCT PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-SRIINSDELABS PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-SRIMASSDELETE PIC S9(9) COMP-4 VALUE +Ø.
Ø2 RS-TOTALEXTENTS PIC S9(4) COMP-4 VALUE +Ø.

*****************************************************************
* OUTPUT INDICATOR
*****************************************************************
Ø1 DS-OUTPUT-2-TABLE.
  Ø2 DS-OUT2-IND PIC S9(4) COMP OCCURS 38 TIMES.

*****************************************************************
* FIELDS FOR RECEIVING
*****************************************************************
Ø1 DS-DB2-LOC-NAME PIC X(16) VALUE 'DB2LOC'.
Ø1 DS-DB2-COMMAND PIC X(8) VALUE SPACES.

*****************************************************************
* DECLARE A RESULT SET LOCATOR FOR THE RESULT SET
* THAT IS RETURNED
*****************************************************************
### SAMPLE RUN

1

***** RTS OBJECT MANAGER FOR DB2 V1R1.00 *****  
COPYRIGHT (C) 2002 - 2003 Author: Lijun Gao. ALL RIGHTS RESERVED.

```
.DSGN001I Job execution starting at 09/22/2003 14:25:51 ...  
.DSGN002I MVS=SP7.0.3,PID=HBB7706,DFSMS=1.3.0,DB2=7.1.0  
.DSGN018I Connected to subsystem DSN3  
.DSGN015I QueryType = RUNSTATS  
.DSGN015I ObjectType = ALL  
.DSGN015I SRTINSDELPCT = 20  
.DSGN015I SRTINSDELABS = 0  
.DSGN015I SRTMASSDELETE = 0  
.DSGN015I SRIINSDELPCT = 20  
.DSGN015I SRIINSDELABS = 0  
.DSGN015I SRIMASSDELETE = 0  
.DSGN015I QUERYSCOPE = WHERE DBNAME LIKE 'TDBECN%' AND DBNAME NOT LIKE 'DSN%'
```

******************************************************************************
*SRIDU: The ratio of the number of INS, UPD, DEL to total number of rows  *
* SRIDA: The number INS, UPD and DEL since the last RUNSTATS  *
* SRMDL: The number of mass deletes since last REORG or LOAD REPLACE  *
******************************************************************************

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>NAME</th>
<th>TP</th>
<th>SRIDU</th>
<th>SRIDA</th>
<th>SRMDL</th>
<th>RUNSTAT-LASTTIME</th>
<th>PART</th>
<th>ASSOC-TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDBECN01</td>
<td>TECNCTE</td>
<td>TS</td>
<td></td>
<td>00733</td>
<td>00003</td>
<td>2003-03-24-14.08.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBECN02</td>
<td>TECNACL</td>
<td>TS</td>
<td></td>
<td>00051</td>
<td>00002</td>
<td>2003-03-24-14.08.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBECN01</td>
<td>TECNCRS</td>
<td>TS</td>
<td></td>
<td>00003</td>
<td>2003-03-24-14.08.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBECN01</td>
<td>XECNTEV3</td>
<td>IX</td>
<td>00091</td>
<td>02777</td>
<td>00003</td>
<td>2003-02-15-09.01.59 TECNTEV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBECN02</td>
<td>XECNBPR1</td>
<td>IX</td>
<td>62876</td>
<td>00063</td>
<td>2003-02-15-09.13.20 TECNBPR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBECN02</td>
<td>XECNBPR2</td>
<td>IX</td>
<td>62876</td>
<td>00063</td>
<td>2003-02-15-09.13.20 TECNBPR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

.DSGN021I TOTAL 6 RECORDS DISPLAYED.

---

Lijun Gao (legend_gao@yahoo.com)
Senior DB2 System Programmer (USA)
© Xephon 2004
Symtrax has announced Release 1.5 of StarQuery, its analytical tool that can create reports from DB2, and many other databases, and send the data straight into Excel for end-user analysis. Also newly available is the StarQuery Runtime module. Runtime users may run queries but cannot modify them, which is ideal for non-technical users who do not need to create their own queries.

StarQuery is split into three modules – StarQuery MapDesigner, StarQuery for Excel, and StarQuery Runtime.

For further information contact: Symtrax, 5777 W Century Blvd, Suite 1745, Los Angeles, CA 90045, USA. Tel: (310) 216 9536. URL: http://www.symtrax.com/en/products/starquery/default.asp.

Veritas Software has announced Version 5.1 of NetBackup, which backs up and recovers files. The product includes extended support for DB2, SQL Server, and Microsoft Exchange.

NetBackup 5.1 has been designed to protect critical data while simplifying the management of back-up and recovery. NetApp snapshot capabilities have been incorporated in the software to create a point-in-time image and NetApp SnapRestore will enable instant single file or entire volume recovery. An advanced client option has been added in Version 5.1 that enables protection for network-attached storage (NAS) environments.

NetBackup 5.1 provides up to 256-bit encryption functionality. Its snapshot techniques are consolidated and leveraged into one simple offering that helps organizations select an appropriate snapshot back-up and recovery method for their specific IT environment. The advanced client extends snapshot capabilities to support DB2, SQL, and Exchange agents.

For further information contact: Veritas, 350 Ellis Street, Mountain View, CA 94043, USA. Tel: (650) 527 8000. URL: http://www.veritas.com/Products/www?c=product&refId=2.

Guardium has announced that its SQL Guard database security platform now offers DB2 UDB 8 support. SQL Guard is a network-based non-intrusive database security platform, which provides a continuous real-time window into all network-based database access. It enables organizations to gain information about activity between all internal/external users and corporate databases.

For further information contact: Guardium, 230 Third Ave, Waltham, MA 02451, USA. Tel: (877) 487 9400. URL: http://www.guardium.com/products.html.

Relicore has expanded the functionality of its Clarity automated application configuration management tool by adding application process discovery capabilities. The AppSense technology in Release 3.5 can automatically discover all running processes on an application server and create a comprehensive, server-to-server dependency map. The product now works with DB2.

For further information contact: Relicore, One Burlington Woods, Burlington, MA 01803, USA. Tel: (781) 229 1122. URL: http://www.relicore.com/products/index.shtml.