88

DB2

February 2000

In this issue

update

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: trevore@xephon.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

Articles published in *DB2 Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

DB2 Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

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.

Printed in England.

DB2 and the Procedural DBA

In 1995 I coined the term *Procedural DBA* to describe a new type of database administration required by modern database management. The concept is a simple one – a new type of DBA is required to manage the procedural logic that is increasingly being stored in relational database management systems.

Initially, the sole purpose of a DBMS was to store, manage, and access data. Over time, database management systems have evolved by integrating procedural logic in the form of complex triggers, stored procedures, and user-defined functions. This code is tightly coupled to the DBMS. As these features are exploited, management tasks such as administration, design, and tuning, are typically assigned to the current DBA staff by default. But this is not always the best approach. What is required is an expansion of the role of database administration.

THE CLASSIC ROLE OF THE DBA

When a DBMS is implemented appropriately, its use spans the enterprise. Multiple applications, consisting of multiple programs, access and manipulate data stored in databases that are managed by the DBMS. A scenario such as this is good because it reduces data redundancy and increases data integrity. However, this same situation also effectively places the DBA on call for all the applications of the organization. If the database portion of any application fails, the DBA must be able to fix the problem, bringing the database back on-line so the application can execute.

To make matters more difficult, the role of the DBA has expanded over the years. In pre-relational days, both database design and data access was complex. Programmers were required to explicitly code program logic to navigate through the database structure to access data. Usually the pre-relational DBA was assigned the task of designing the hierarchic or network database design. Almost always, this process consisted of both logical and physical database design, although it was not always recognized as such at the time. Once the database was planned, designed, and generated, and the DBA created back-up and

recovery jobs, little more than space management and reorganizations were required. Of course, this sounds easier than it actually was. Pre-relational DBMS products (such as IMS) require a complex series of utility programs to be run in order to perform back-up, recovery, and reorganization, consuming a large amount of time and effort.

Today, of course, DBAs still design databases and perform tasks such as back-up, recovery, and reorganization. But increasingly databases are generated from logical data models created by data administration staff using data modelling and database design tools. Additionally, the utilities for performing back-up, recovery, and reorganization are simpler to build in the relational world.

Although the up-front effort required to design a relational database is reduced, it is not eliminated. Relational design still requires physical implementation decisions such as table design, partitioning, indexing, normalization, and denormalization. But instead of just performing physical implementation and administration, DBAs are more intimately involved with procedural data access too. The nature of relational technology requires additional involvement during the design of data access routines. This is true because relational optimizer technology embedded in the RDBMS is used to choose the best access paths to the data. The optimization choices must be reviewed by the DBA. Therefore, application program and SQL design reviews are a vital component of the DBA's job.

Furthermore, DBAs perform most monitoring and tuning responsibilities. DBAs use tools like EXPLAIN, performance monitors, and SQL analysis tools to proactively administer RDBMS applications.

Often, DBAs are not adequately trained in these areas. It is a distinctly different skill to program than it is to create well-designed relational databases. Yet, DBAs quickly learn that they have to be able to understand application programming techniques to succeed.

DBMS-COUPLED APPLICATION LOGIC

Although DB2 was one of the last major RDBMS products to gain a full complement of tools for storing procedural logic in the database,

its current support as of Version 6 is very robust. DB2 provides support for stored procedures, triggers, user-defined functions, a procedural version of SQL based on SQL/PSM, and user-defined data types.

A procedural SQL language adds features such as looping, branching, and flow of control statements to make SQL a more functionally complete and useful programming language. Using DB2's version of SQL/PSM, developers can create complex functional stored procedures and triggers without the need to code a 3GL program.

Let's define the different types of logic that can be stored, accessed, and managed in DB2 databases.

Stored procedures are procedural logic that is maintained, administered, and executed through the RDBMS. The primary reason for using stored procedures is to move application code off the client and on to the database server. This can result in reduced overhead because one client can invoke a stored procedure consisting of multiple SQL statements. Invoking one procedure to execute multiple SQL statements is preferable to the client executing multiple SQL statements directly because it minimizes network traffic, thereby enhancing overall application performance. A stored procedure is not 'physically' associated with any other object in the database. It can access and/or modify data in one or more tables. Basically, stored procedures can be thought of as 'programs' that 'live' in the RDBMS.

Triggers are event-driven specialized procedures that are stored in, and executed by, the RDBMS. Each trigger is attached to a single, specified table. Triggers can be thought of as an advanced form of 'rule' or 'constraint' written using procedural logic. A trigger cannot be directly called or executed; it is automatically executed (or 'fired') by the RDBMS as the result of an action – usually a data modification to the associated table. Once a trigger is created, it is always executed when its 'firing' event occurs (update, insert, delete, etc).

User-defined functions (UDFs) provide developers with the ability to extend the SQL language. Once coded, a UDF can be specified wherever a built-in SQL function can be specified. In general, DB2

functions (both built-in and user-defined) can be used anywhere an expression can be used (with some exceptions). Functions are called by specifying the function name and any required operands.

You can think of stored procedures and triggers and user-defined functions in the same way as other database objects such as tables, views, and indexes, because they are controlled by and managed within DB2. These objects are often collectively referred to as server code objects (SCOs), because they are actually program code that is managed by a database server as a database object.

WHY USE SERVER CODE LOGIC?

The predominant reason for using SCOs is to promote code reusability. Instead of replicating code on multiple servers or within multiple application programs, SCOs enable code to reside in a single place – the database server. SCOs can be automatically executed, based on context and activity, or can be called from multiple client programs as required. This is preferable to cannibalizing sections of program code for each new application that must be developed. SCOs enable logic to be invoked from multiple processes instead of being re-coded into each new process every time the code is required.

An additional benefit of SCOs is increased consistency. If every user and every database activity (with the same requirements) is assured of using the SCO instead of multiple replicated code segments, then the organization can be assured that everyone is running the same, consistent code. If each individual user deployed his or her own individual and separate code, no assurance could be given that the same business logic was being used by everyone. In fact, it is almost a certainty that inconsistencies would occur.

Additionally, SCOs are useful for reducing the overall code maintenance effort. Because SCOs exist in a single place (the RDBMS), changes can be made quickly without requiring propagation of the change to multiple workstations.

Finally, SCOs can be coded to support database integrity constraints, implement security requirements, reduce code maintenance efforts, support remote data access, and, as mentioned earlier, enhance

performance. Of course, in order to achieve these gains, SCOs need to be effectively managed and administered. Hence the need for a Procedural DBA.

THE PROCEDURAL DBA

Once server code objects are coded and made available to the RDBMS, applications and developers will begin to rely upon them. Although the functionality provided by SCOs is unquestionably useful and desirable, DBAs are presented with a major dilemma. Now that procedural logic is being stored in DB2, DBAs must grapple with the issues of quality, maintainability, and availability. How and when will these objects be tested? The impact of a failure is enterprise-wide, not relegated to a single application. This increases the visibility and criticality of these objects. Who is responsible if they fail? The answer should be – a DBA. But testing and debugging of code is not a typical role for DBAs.

With the advent of server code objects, the role of the DBA is expanding to encompass too many responsibilities for a single person to perform the job capably. The solution is to split the job of DBA into two separate parts, based upon the type of database object being supported – data objects or server code objects.

Administering and managing data objects is more in line with the traditional role of the DBA, and is well-defined. But DDL and database utility experts cannot be expected to debug procedures and triggers written in sometimes very complex SQL and application code. Debugging a procedure is a very different task from creating a database schema and ensuring that there are no syntax errors. Furthermore, even though many organizations rely upon DBAs to be the SQL experts in the company, often they are not – at least not DML experts. Simply because DBAs know the best way to create a physical database design and DDL does not mean they will know the best way to access that data. It is not uncommon for a DBA to come from the ranks of network and/or system administration, and SQL is foreign to many system adminstrators.

The role of administering the procedural logic in an RDBMS should

fall upon someone skilled in that discipline. A new type of DBA must be defined to accommodate server code object and procedural logic administration. This new role can be defined as a Procedural DBA.

The Procedural DBA should be responsible for database management activities that require programming and similar activities. This includes primary responsibility for server code objects. Whether SCOs are actually programmed by the Procedural DBA may differ from shop to shop. This will depend on the size of the shop, the number of DBAs available, and the scope of server code object implementation. Minimally, the Procedural DBA should lead SCO code reviews and perform SCO administration.

Additionally, the Procedural DBA must be on call for SCO failures. Consider the ramifications if this were not the case. As part of a new project, Team A implements a new stored procedure to obtain customer information. The project is completed and moved to production. Team B, working in a different department, decides to re-use the customer information stored procedure in their new application. Team B completes its project and moves it into production. Three weeks later, at 2:00am, Team B's application fails because of a failure in the shared stored procedure. Who comes in to fix it? Team A, the team that coded the stored procedure but whose application is working? Or Team B, whose application is down but has no-one who understands the stored procedure?

Of course, the correct answer is the Procedural DBA.

Additional issues arise as SCOs are implemented. One such example is the proper maintenance of DB2 triggers once they have been defined. If multiple triggers are defined on the same table for the same action, DB2 will fire the triggers in the order they were created. This can have an impact on the end result of a transaction. Consider, for example, the following – admittedly contrived – scenario. Two update triggers exist on TABLE1, namely TRIGGER1 and TRIGGER2. TRIGGER1 adds the value 5 to COLX in TABLE2; TRIGGER2 multiplies the value of COLX in TABLE2 by the value 3. If COLX is equal to 3, after the two triggers fire, COLX would be equal to (3 + 5) * 3, or 24.

Now, consider what would happen if TRIGGER1 is modified for some reason. The trigger is dropped and recreated. Now, TRIGGER2 would fire first, followed by TRIGGER1. The following example would have a different result. If COLX is equal to 3, after the two triggers fire, COLX would be equal to (3 * 3) + 5, or 14. The order in which triggers fire is important, but the only way to control it is based on which trigger was created first. This process must be controlled by someone who understands these nuances, ideally a Procedural DBA.

Other procedural administrative functions that can be allocated to the Procedural DBA include application code reviews, access path review and analysis, SQL debugging, complex SQL analysis, and re-writing queries for optimal execution. Off-loading these tasks to the Procedural DBA will enable the traditional, data-oriented DBAs to concentrate on the actual physical design and implementation of databases. This should result in much better designed databases, and much better performing SQL.

The Procedural DBA should still report through the same management unit as the traditional DBA and not through the application programming staff. This enables better skills sharing between the two distinct DBA types. Of course, there will need to be a greater synergy between the procedural DBA and the application programmer/analyst. In fact, the typical job path for the procedural DBA should come from the application programming ranks because this is where the coding skill-base exists.

SYNOPSIS

As you implement triggers, stored procedures, and user-defined functions to support business rules in your applications, be aware that database administration becomes more complex. The role of the DBA is rapidly expanding to the point where no single professional can be reasonably expected to be an expert in all facets of the job. It is high time that the job is explicitly defined into manageable components, starting with the Procedural DBA.

Craig S Mullins
Director, DB2 Technology Planning
BMC Software (USA)

Optimizing tablespace and indexspace – part 2

This month we conclude the code to determine the current space requirements.

```
if substr(row.1,2) = 'NO CATALOG ENTRIES FOUND' then do
   Call Free_proc
   Call Aloc
   address ispexec 'tbend messdb'
   Call Create_messg
   message = 'No catalog entries found, check Search Fields.'
   Call Error 'crec'
end
else do
   address ispexec 'addpop row(1) column(5)'
   address ispexec 'tbcreate "blist" names(v1 v2 v3 v4 v5 v6)'
   count=Ø
   num=row.Ø
   do i=1 to row.Ø
      if substr(row.i,2,1)='B' then do
         v6=' '
         v1= word(row.i,2)
         v2= word(row.i,3)
         v3= word(row.i,4)
         v4= word(row.i,5)
         v5= right(word(row.i,6),13)
         if v5 = '-1' then v6='Runstat'
         address ispexec 'tbadd "blist"'
      end
      else count=count+1
   end
   address ispexec 'tbtop "blist"';
   address ispexec 'tbdispl "blist" panel(ALTTSL)'
   if rc=8 then do
      Call Free_proc
      address ispexec 'tbend "blist"'
      Call Aloc
      address ispexec rempop all
      address ispexec 'tbend messdb'
      Call Create_messg
      signal top
   end
end
Call Free_proc
ctime=time('s')
messg = 'Reading LISTCAT Information'
messg = time() || " " || messg
Call Send_messg
```

```
light='OFF'
address ispexec 'tbcreate,
  "tlist" names(scu db cr ts ix ob pr pri sec a1 a2 a3 a4 a5 a6 dt)'
asterisks= '************************************************'
procent=100/(2*count)
j=0
scu=0
do i=1 to row.0
   if (word(row.i,1)='TS') & (obj='TS' | obj='BOTH') then do
      scu=scu+1
      db = word(row.i,2)
      ts = word(row.i,3)
      pr = word(row.i,4)
      ix = db
      ob='TS'
      a1 = 'TS:'||left(db||'.'||word(row.i,3),18)||,
           right(word(row.i,4),3)
      a4 = right(word(row.i,5),8)
      a5 = right(word(row.i,6),7)
      Call Logic
      a2 = right(a2,8)
      a3 = right(a3,7)
      dt='TS:'||right(word(row.i,2),10)||'  '||,
         left(word(row.i,3),29)||right(word(row.i,5),11)||,
         right(word(row.i,6),9)
      Call Add
   end
   if (word(row.i,1)='IX') & (obj='IX' | obj='BOTH') then do
      scu=0
      db = word(row.i,2)
      cr = word(row.i,3)
      ts = word(row.i,8)
      ix = word(row.i,4)
      ob='IX'
      pr = word(row.i,5)
      a1 = 'IX:'||left(cr||'.'||word(row.i,8),18)||,
           right(word(row.i,5),3)
      a4 = right(word(row.i,6),8)
      a5 = right(word(row.i,7),7)
      Call Logic
      pri= 0
      sec= 0
      a2 = right(a2,8)
      a3 = right(a3,7)
      crix=word(row.i,3)||':'||word(row.i,8)
      dt='IX:'||right(word(row.i,2),10)||'  '||left(crix,29)||,
          right(word(row.i,6),11)||right(word(row.i,7),9)
      Call Add
   end
end
```

```
if light='OFF' then do
   Call Free_proc
   address ispexec 'tbend "blist"'
   address ispexec 'tbend "tlist"'
   Call Aloc
   address ispexec rempop all
   address ispexec 'tbend messdb'
   Call Create_messg
   message = 'No entries found for option '||op||'.'
   Call Error 'crec'
end
ctime=(time('s')-ctime)%60 min (time('s')-ctime)//60
address ispexec 'tbtop "tlist"'
address ispexec 'tbdispl "tlist" panel(ALTTSR)'
if rc=8 then do
   Call Free_proc
   address ispexec 'tbend "blist"'
   address ispexec 'tbend "tlist"'
   Call Aloc
   address ispexec rempop all
   address ispexec 'tbend messdb'
   Call Create_messg
   signal top
end
messg = 'Building a ALTER TS/IX JCL'
messg = time() || " " || messg
Call Send_messg
/* JCL Skeleton DB2 ALTER TS/IX Utility    */
if vol  = ' '
then ivol =0
else ivol =1
if vola = ' '
then ivola=0
else ivola=1
if volb = ' '
then ivolb=0
else ivolb=1
address ispexec rempop all
tsufA=time('M')+1
dsufB='D'||right(date('D'),3,'0')||right(time('M'),4,'0')
dsufA='D'||right(date('D'),3,'0')||right(tsufA,4,'0')
date=date()
time=time(c)
user=userid()
tempfile=userid()||'.UTIL.ALTER.JCL'
address tso
"delete '"tempfile"'"
"free dsname('"tempfile"')"
"free ddname(ispfile)"
"free attrlist(formfile)"
```

```
"attrib formfile blksize(800) lrecl(80) recfm(f b) dsorg(ps)"
"alloc ddname(ispfile) dsname('"tempfile"')",
       "new using (formfile) unit(3390) space(1 1) cylinders"
address ispexec
"ftopen"
"ftincl ALTTSIX"
"ftclose"
zedsmsg = "JCL shown"
zedlmsg = "JCL ALTER TS/IX shown"
"setmsg msg(isrz001)"
"edit dataset('"tempfile"')"
address ispexec 'tbend "tlist"'
address ispexec 'tbend "blist"'
address ispexec  "tbclose "messdb""
exit
Aloc:
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".UTIL.ALTER'"
  "ALLOC DD(SYSPRINT) DSN('"SYSVAR(SYSUID)".UTIL.ALTER') SPACE(24 8),
  TRACK MOD UNIT(3390) RECFM(F,B) LRECL(90) BLKSIZE(900) ,
  F(SYSPRINT) CATALOG REUSE "
Return
Error:
  ARG cur_par
  cur=cur_par
  address ispexec "setmsg msg(alt001)"
  signal top
Return
Free_proc:
  "execio 0 diskr sysprint (finis"
  address tso "free f(sysprint)"
Return
Logic:
  w=3
  if word(row.i,1)='IX' then w=4
  j=j+1
  messg = substr(asterisks,1,trunc(procent*j,0))
  Call Send_messg
  suf= word(row.i,w+1)
  if suf=0 then suf=1
  if word(row.i,1)='IX'
  then catn=word(row.i,9)
  else catn=word(row.i,7)
  part='I0001.A'||right(suf,3,'0')
  file=catn||'.DSNDBD.'||db||'.'||ts||'.'||part
  dsn = "('"file"')"
  X=OUTTRAP('var.')
  address tso "listc" entries dsn allocation
  X=OUTTRAP('OFF')
  Call Check_dsn
  harba = word(translate(var.8,' ','-'),7)
```

```
  hurba = word(translate(var.9,' ','-'),7)
  if hurba < trunc(737280/trk,0) then do
     pri=1
     sec=1
  end
  else do
     pri=trunc((hurba/(737280/trk)+1),0)
     sec=max(trunc(pri*0.05,0),1)
  end
  ext   = word(translate(var.12,' ','-'),12)
  if trunc(harba/1000,0) < 12 then do
     a2=12
     a3=12
  end
  else do
     a2=trunc(hurba/1000,0)
     a3=trunc(a2/7,0)
  end
  a6=' '
  pro=TRUNC(ABS(a4/a2-1)*100)
  a4p=a4
  if a4p=0 then a4p=1
  ods=ABS(1-a2/a4p)
  if (ods<0.002 | a2=a4 | a2>a4) & a2<(a4+a5) then a6='No Comment'
  if a2*1.05 < a4 then a6='Ov-All '||right(pro,4)||'%'
  if ext > 1 then a6='Hi-Ext '||right(ext,5)
Return
Add:
  if op=' ' then do
     address ispexec 'tbadd "tlist"'
     light='ON'
  end
  else do
     if op='E' & ext>1 then do
        address ispexec 'tbadd "tlist"'
        light='ON'
     end
     if op='O' & substr(a6,1,2)='Ov' then do
        if pct<>' ' then do
           if pro>pct-1 then do
              address ispexec 'tbadd "tlist"'
              light='ON'
           end
        end
        else do
           address ispexec 'tbadd "tlist"'
           light='ON'
        end
     end
  end
```

```
 Return
 Check_dsn:
  if rc>Ø then do
     message=file||' not found.'
     address ispexec "setmsg msg(altØØ1)"
     Call Free_proc
     Call Aloc
     address ispexec 'tbend "blist"'
     address ispexec 'tbend "tlist"'
     address ispexec  "tbclose "messdb""
     Call Create_messg
     signal top
  end
 Return
 Create_messg:
  messg = "s"||userid()
  address ispexec "tbcreate "messdb" names(messg) write replace"
 Return
 Send_messg:
  address ispexec "tbadd " messdb
  address ispexec "control display lock "
  address ispexec "addpop row(13) column(6)"
  address ispexec "tbdispl "messdb" panel(altmes)"
  address ispexec rempop
 Return
```

ALTTSM

```
)Attr Default(%+_)
   ! type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
!-/-/- ? ALTER Space - Parameter Entry +!-/-/-
%Command ===>_zcmd
+
+
#PARAMETER  #PARAMETER VALUE                #PROMPT                    +
+
+SSID     =>[db2 +                          DB2 Sub-System Identifier
+Object   =>[obj +                          TS,IX or BOTH
+Option   =>[op+   [pct +                   E-Extent,O-Over Allocation
+Creator  =>[crec    +                      Table Creator
+Name     =>[tabc             +             Table Name
+Tsname   =>[tsnc    +                      Tablespace Name
+Dbname   =>[dbnc    +                      Database   Name
```

```
+Icopy    =>[ico   +                          Before-After-Both-None
+Copypref =>[pref   +                         Dataset PREFIX for ICs
+Shrlevel =>[rtype    +                       None-Reference-Change
+Log      =>[log+                             Log option!YES+or!NO
+Sortdata =>[sor+                             Sortdata!YES+or!NO
+Keepdict =>[dic+                             Keepdictionary!YES+or!NO
+Runstats =>[rru+                             Runstats tspace !YES+or!NO
+Volume   =>[vol   + [vola  + [volb  +        Volser
+Tracks   =>[trk+                             Tracks per Cylinder - 15
+
            $msg
+
                                  } PF3 Return +
)Init
  if (&db2 ¬= ' ')
      .attr (db2) = 'pad(nulls)'
  if (&op  ¬= ' ')
      .attr (op) = 'pad(nulls)'
  if (&pct ¬= ' ')
      .attr (pct) = 'pad(nulls)'
  if (&obj ¬= ' ')
      .attr (obj) = 'pad(nulls)'
  if (&crec ¬= ' ')
      .attr (crec) = 'pad(nulls)'
  if (&tabc ¬= ' ')
      .attr (tabc) = 'pad(nulls)'
  if (&tsnc ¬= ' ')
      .attr (tsnc) = 'pad(nulls)'
  if (&dbnc ¬= ' ')
      .attr (dbnc) = 'pad(nulls)'
  if (&ico ¬= ' ')
      .attr (ico) = 'pad(nulls)'
  if (&pref ¬= ' ')
      .attr (pref) = 'pad(nulls)'
  if (&rtype¬= ' ')
      .attr (rtype) = 'pad(nulls)'
  if (&log ¬= ' ')
      .attr (log) = 'pad(nulls)'
  if (&sor ¬= ' ')
      .attr (sor) = 'pad(nulls)'
  if (&rru ¬= ' ')
      .attr (rru) = 'pad(nulls)'
  if (&dic ¬= ' ')
      .attr (dic) = 'pad(nulls)'
  if (&vol ¬= ' ')
      .attr (vol) = 'pad(nulls)'
  if (&vola¬= ' ')
      .attr (vola) = 'pad(nulls)'
  if (&volb¬= ' ')
      .attr (volb) = 'pad(nulls)'
```

```
  if (&trk ¬= ' ')
      .attr (trk) = 'pad(nulls)'
  &msg='Enter values for the ALTER TS/IX space service !'
)Reinit
)Proc
  &sr = TRUNC(&obj,' ')
  if (&obj='T') &obj = 'TS'
  if (&obj='I') &obj = 'IX'
  if (&obj='B') &obj = 'BOTH'
  &sr = TRUNC(&ico,' ')
  if (&sr='A' ! &sr='AF') &ico = 'AFTER'
  if (&sr='B' ! &sr='BE') &ico = 'BEFORE'
  if (          &sr='BO') &ico = 'BOTH'
  if (&sr='N' ! &sr='NO') &ico = 'NONE'
  &sr = TRUNC(&rtype,' ')
  if (&sr='N' ! &sr='NO') &rtype = 'NONE'
  if (&sr='R' ! &sr='RE') &rtype = 'REFERENCE'
  if (&sr='C' ! &sr='CH') &rtype = 'CHANGE'
  VPUT (db2 obj crec tabc tsnc dbnc ico pref)    PROFILE
  VPUT (rtype log sor rru dic vol vola volb trk) PROFILE
)End
```

ALTTSL

```
)Attr Default(%+_)
   ( type(text  ) intens(high) hilite(reverse)
   ] type(text  ) intens(high) hilite(reverse) color(green)
   / type(text  ) intens(high) hilite(reverse) color(yellow)
   ~ type(output) intens(high)                 color(red)
   [ type(output) intens(high) hilite(reverse) color(green) caps(off)
   + type(text  ) intens(low )
   _ type( input) intens(high) caps(on ) just(left )
   ¬ type(output) intens(low ) caps(off) just(asis )
)Body window(74,19)
/ ALTER Space Service - Selection Result +
+
+Command ===>_zcmd                                  +Scroll ===>_amt +
+
+Press]Enter+to have this service continue.
+Press]End  +to respecify your  PARAMETERS.
+
]Dbname  +]Tsname  +]Table          +]Creator +       ]  Card+/Message+
+
)Model
¬z       +¬z       +¬z              +¬z       +¬z            +~z      +
)Init
  &ZWINTTL = 'Alter table-index space'
  .ZVARS = '(v1 v2 v3 v4 v5 v6)'
  &amt = PAGE
```

```
)Reinit
)Proc
)End
```

ALTTSR

```
)Attr Default(%+_)
   ( type(text  ) intens(high) hilite(reverse)
   ] type(text  ) intens(high) hilite(reverse) color(green)
   / type(text  ) intens(high) hilite(reverse) color(yellow)
   } type(text  ) intens(high) hilite(reverse) color(red)
   { type(text  ) intens(high) hilite(reverse) color(white)
   ~ type(output) intens(high)                 color(red)
   # type(output) intens(high)                 color(yellow)
   [ type(output) intens(high) hilite(reverse) color(green) caps(off)
   + type(text  ) intens(low )
   _ type( input) intens(high) caps(on ) just(left )
   ¬ type(output) intens(low ) caps(off) just(asis )
)Body
/ ALTER Space Analysis Recommendation Panel +
+
+Command ===>_zcmd                                    +Scroll ===>_amt +
+
+Press]Enter+to have this service continue.
+Press]End  +to respecify your  PARAMETERS.
+
                        } Recommendation ]Current allocation+
{Object       Part}PRIQTY  }SECQTY ]PRIQTY  +  ]SECQTY/Message        +
+
)Model
¬z               #z       #z      ¬z       + ¬z      ~z              +
)Init
  &ZWINTTL = 'Analysis panel'
  .ZVARS = '(a1 a2 a3 a4 a5 a6)'
  &amt = PAGE
)Reinit
)Proc
)End
```

ALTMES

```
)attr default($+_)
| type (text)   intens(low)  color(white)
@ type (text)   intens(high) color(red)   caps(off) hilite(reverse)
! type (input)  intens(non)  color(green) caps(on)  just(left)
# type (output) intens(high) color(yellow)  caps(off)
)body default($~\) window(53,3)
!zcmd +     @ Message display !amt  |
```

```
|————————50%——————100% |
)model clear(messg)
#z                                                  +
)init
     .zvars = '(messg)'
)reinit
)proc
  if (.pfkey = pf03) &pf3 = exit
)end
```

ALT00

```
ALT001           .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

PALTTSI

```
* PROCESS GS,OFFSET,OPT(TIME);
 PALTTSI:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*****************************************************************/
 /* DESCRIPTION: PL/I PROGRAM FOR ALTER SPACE SERVICE             */
 /*****************************************************************/
  DCL PARMS CHAR(100) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(0);
  DCL HCARD         BIN FIXED(31);
  DCL MCARD         PIC'-.-.-.-9';
  DCL HTBNAME       CHAR(18);
  DCL HTBCREATOR    CHAR(8);
  DCL IXCREATOR     CHAR(8);
  DCL IXSPACE       CHAR(8);
  DCL HDBNAME       CHAR(8);
  DCL HTSNAME       CHAR(8);
  DCL HIXNAME       CHAR(18);
  DCL HPART         BIN FIXED(15);
  DCL HPQTY         BIN FIXED(31);
  DCL HSQTY         BIN FIXED(31);
  DCL PQTY          PIC'ZZZZZZZ9';
  DCL SQTY          PIC'ZZZZZZZ9';
  DCL PART          PIC'ZZ9';
  DCL VCAT          CHAR(8);
  DCL 1 WORKST,
      2 CREC        CHAR(8)  VAR,
      2 TABC        CHAR(18) VAR,
      2 TSNC        CHAR(8)  VAR,
      2 DBNC        CHAR(8)  VAR,
      2 OBJ         CHAR(4);
```

```
DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
DCL IC            BIN FIXED(15);
DCL OUT           CHAR(18) VAR;
EXEC SQL INCLUDE SQLCA;
IF SUBSTR(PARMS,1,8)=' ' THEN CREC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,1,8),OUT);
   CREC=OUT;
   IF LENGTH(CREC) < 8 THEN CREC=CREC||'%';
END;
IF SUBSTR(PARMS,9,18)=' ' THEN TABC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,9,18),OUT);
   TABC=OUT;
   IF LENGTH(TABC) < 18 THEN TABC=TABC||'%';
END;
IF SUBSTR(PARMS,27,8)=' ' THEN TSNC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,27,8),OUT);
   TSNC=OUT;
   IF LENGTH(TSNC) < 8 THEN TSNC=TSNC||'%';
END;
IF SUBSTR(PARMS,35,8)=' ' THEN DBNC='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,35,8),OUT);
   DBNC=OUT;
   IF LENGTH(DBNC) < 8 THEN DBNC=DBNC||'%';
END;
OBJ=SUBSTR(PARMS,43,4);

/* SELECTION RESULTS                                */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
DBNAME, TSNAME, CREATOR, NAME, CARD
FROM SYSIBM.SYSTABLES
WHERE CREATOR LIKE :CREC
  AND NAME    LIKE :TABC
  AND TSNAME  LIKE :TSNC
  AND DBNAME  LIKE :DBNC
  AND TYPE = 'T'
ORDER BY CREATOR,NAME
FOR FETCH ONLY;
EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO
     :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
IF HTBCREATOR='SYSIBM'
THEN DO;
   PUT SKIP LIST ('DB2 CATALOG');
   GOTO VEN;
END;
```

```
DO WHILE (SQLCODE=Ø);
   NUMSEQ=1;
   MCARD=HCARD;
   PUT SKIP LIST ('B '||HDBNAME||' '||HTSNAME||' '||
            SUBSTR(HTBNAME,1,18)||' '||HTBCREATOR||' '||MCARD);
   EXEC SQL FETCH C1 INTO
        :HDBNAME, :HTSNAME, :HTBCREATOR, :HTBNAME, :HCARD;
END;
EXEC SQL CLOSE C1;
IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');

/* TABLESPACE ALLOCATION                          */
IF OBJ='TS' | OBJ='BOTH' THEN DO;
   EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR SELECT
   DISTINCT P.DBNAME,P.TSNAME,PARTITION,
   PQTY*PGSIZE,SQTY*PGSIZE,P.VCATNAME
   FROM SYSIBM.SYSTABLESPACE S,
        SYSIBM.SYSTABLEPART  P,
        SYSIBM.SYSTABLES T
   WHERE T.CREATOR LIKE :CREC
     AND T.NAME    LIKE :TABC
     AND T.TSNAME  LIKE :TSNC
     AND T.DBNAME  LIKE :DBNC
     AND T.TYPE = 'T'
     AND S.DBNAME  = P.DBNAME
     AND S.NAME    = P.TSNAME
     AND T.DBNAME  = S.DBNAME
     AND T.TSNAME  = S.NAME
   ORDER BY P.TSNAME,PARTITION
   FOR FETCH ONLY;
   EXEC SQL OPEN C2;

   EXEC SQL FETCH C2 INTO
        :HDBNAME, :HTSNAME, :HPART, :HPQTY, :HSQTY, :VCAT;
   DO WHILE (SQLCODE=Ø);
      PART=HPART;
      PQTY=HPQTY;
      SQTY=HSQTY;
      PUT SKIP LIST ('TS '||HDBNAME||' '||HTSNAME
               ||' '||HPART||' '||PQTY||' '||SQTY||' '||VCAT);
      EXEC SQL FETCH C2 INTO
           :HDBNAME, :HTSNAME, :HPART, :HPQTY, :HSQTY, :VCAT;
   END;
   EXEC SQL CLOSE C2;
END;

/* INDEXSPACE ALLOCATION                          */
IF OBJ='IX' | OBJ='BOTH' THEN DO;
   EXEC SQL DECLARE C3 CURSOR WITH HOLD FOR SELECT
   DISTINCT I.CREATOR,I.DBNAME,P.IXNAME,
```

```
    PARTITION,PQTY*4,SQTY*4,I.INDEXSPACE,P.VCATNAME
    FROM SYSIBM.SYSINDEXES I,
         SYSIBM.SYSINDEXPART P,
         SYSIBM.SYSTABLES T
    WHERE T.CREATOR LIKE :CREC
      AND T.NAME    LIKE :TABC
      AND T.TSNAME  LIKE :TSNC
      AND T.DBNAME  LIKE :DBNC
      AND TYPE = 'T'
      AND I.NAME    = P.IXNAME
      AND I.CREATOR = P.IXCREATOR
      AND T.CREATOR = I.TBCREATOR
      AND T.NAME    = I.TBNAME
    ORDER BY P.IXNAME,PARTITION
    FOR FETCH ONLY;
    EXEC SQL OPEN C3;

    EXEC SQL FETCH C3 INTO :IXCREATOR,:HDBNAME,
         :HIXNAME,:HPART,:HPQTY,:HSQTY,:IXSPACE,:VCAT;
    DO WHILE (SQLCODE=Ø);
       PART=HPART;
       PQTY=HPQTY;
       SQTY=HSQTY;
       PUT SKIP LIST ('IX '||HDBNAME||' '||IXCREATOR||' '||HIXNAME
        ||' '||HPART||' '||PQTY||' '||SQTY||' '||IXSPACE||' '||VCAT);
       EXEC SQL FETCH C3 INTO :IXCREATOR,:HDBNAME,
            :HIXNAME,:HPART,:HPQTY,:HSQTY,:IXSPACE,:VCAT;
    END;
    EXEC SQL CLOSE C3;
  END;

  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
  VEN:
 END PALTTSI;
```

ALTTSIX

```
)TBA 72
)CM ————————————————————————————————
)CM Skeleton to generate JCL for ALTER tablespace/indexspace        –
)CM ————————————————————————————————
//&user.X JOB (12ØØ-12Ø5-ØØ),'&option',
//             NOTIFY=&user,REGION=4M,
```

```
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ********************************************************************
//*    ALTER SPACE service
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    CALCULATING TIME IS &ctime SECONDS.
//*
//*    ALTER - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    _____  ___________
//*    SSID     :  &db2
//*    Object   :  &obj
//*    Option   :  &op   &pct
//*    Creator  :  &creC
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    Icopy    :  &ico
//*    Copypref :  &pref
//*    Shrlevel :  &rtype
//*    Log      :  &log
//*    Sortdata :  &sor
//*    Keepdict :  &dic
//*    Runstats :  &rru
//*    Volume   :  &vol  &vola  &volb
//*    Catname  :  &catn
//*    Tracks   :  &trk
//* ********************************************************************
//*  NUM  DATABASE  TS/IX SPACE                        PRIQTY   SECQTY
//*  _  ____  ________________  ___  ___
)DOT "TLIST"
//*  &dt
)ENDDOT
//*  _  ____  ________________  ___  ___
//*_______________________________
//*
)SEL &rtype EQ NONE
//*-- STOP TABLESPACES ------------------
//STOPTS   EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)BLANK 1
)DOT "TLIST"
)SEL &pr EQ 0
   -STOP  DATABASE(&db) SPACENAM(&ts)
)ENDSEL
)SEL &pr GT 0
   -STOP  DATABASE(&db) SPACENAM(&ts) PART(&pr)
```

```
)ENDSEL
)ENDDOT
)ENDSEL
//*
//RUNSQL  EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=DSN510.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(&db2)
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP51) -
     LIB('DSN510.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  SET CURRENT SQLID = '&user' ;
)DOT "TLIST"
)SEL &scu GT 0
  ALTER TABLESPACE &db..&ts
        COMPRESS YES
)ENDSEL
)SEL &scu EQ 0
  ALTER INDEX &cr..&ix
)ENDSEL
)SEL &pr GT 0
        PART &pr
)ENDSEL
        PRIQTY &a2     SECQTY &a3       ;
-       PRIQTY &a4     SECQTY &a5       ;  ORIGINAL-VALUES
)ENDDOT
)SEL &rtype EQ NONE
//*-- START TABLESPACES - UTILITY MODE --------
//STARUT   EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)BLANK 1
)DOT "TLIST"
)SEL &pr EQ 0
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(UT)
)ENDSEL
)SEL &pr GT 0
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(UT) PART(&pr)
)ENDSEL
)ENDDOT
)ENDSEL
//*
)SEL &ico = BEFORE | &ico = BOTH
//*-- BEFORE IMAGE COPY  ----------------
```

```
//COPYB    EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//         UID='&user..COPYB',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
)DOT "TLIST"
)SEL &scu GT Ø
)SEL &ivol EQ Ø
//COPB&scu   DD UNIT=SYSDA,
)ENDSEL
)SEL &ivol EQ 1
//COPB&scu   DD UNIT=339Ø,VOL=SER=&vol,
)ENDSEL
)SEL &pr EQ Ø
//        DSN=&pref..&db..&ts..&dsufb,
)ENDSEL
)SEL &pr GT Ø
//        DSN=&pref..&db..&ts..P&pr..&dsufb,
)ENDSEL
//        DCB=(BUFNO=2Ø,BLKSIZE=22528),
//        DISP=(,CATLG,DELETE),
//        SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)ENDDOT
//SYSIN  DD  *
)DOT "TLIST"
)SEL &scu GT Ø
)BLANK 1
  COPY  TABLESPACE &db..&ts
)SEL &pr GT Ø
             DSNUM &pr
)ENDSEL
           COPYDDN COPB&scu
              FULL YES
          SHRLEVEL REFERENCE
)ENDSEL
)ENDDOT
/*
)ENDSEL
//*———————————————————————
//*—— REORG ————————————————
//REORG  EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//       UID='&user..REORG',UTPROC=''
//STEPLIB DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
)SEL &ivol EQ Ø
//SYSREC  DD UNIT=SYSDA,
)ENDSEL
)SEL &ivol EQ 1
//SYSREC  DD UNIT=339Ø,VOL=SER=&vol,
)ENDSEL
//        DSN=&user..REORG.SYSREC.&dsufb,
//        SPACE=(CYL,(1ØØ,1ØØ,),RLSE,,ROUND),
```

```
//        DISP=(NEW,DELETE,CATLG)
)SEL &ivola EQ Ø
//SORTOUT DD UNIT=SYSDA,
)ENDSEL
)SEL &ivola EQ 1
//SORTOUT DD UNIT=339Ø,VOL=SER=&vola,
)ENDSEL
//        DSN=&user..REORG.SORTOUT.&dsufb,
//        SPACE=(CYL,(1ØØ,1ØØ,),RLSE,,ROUND),
//        DISP=(NEW,DELETE,CATLG)
)SEL &ivolb EQ Ø
//SYSUT1  DD UNIT=SYSDA,
)ENDSEL
)SEL &ivolb EQ 1
//SYSUT1  DD UNIT=339Ø,VOL=SER=&volb,
)ENDSEL
//        DSN=&user..REORG.SYSUT1.&dsufb,
//        SPACE=(CYL,(1ØØ,1ØØ,),RLSE,,ROUND),
//        DISP=(NEW,DELETE,CATLG)
)SEL &rtype EQ REFERENCE OR &rtype EQ CHANGE
)DOT "TLIST"
)SEL &scu GT Ø
)SEL &ivol EQ Ø
//COPY&scu   DD UNIT=SYSDA,
)ENDSEL
)SEL &ivol EQ 1
//COPY&scu   DD UNIT=339Ø,VOL=SER=&vol,
)ENDSEL
)SEL &pr EQ Ø
//        DSN=&pref..&db..&ts..&dsufb,
)ENDSEL
)SEL &pr GT Ø
//        DSN=&pref..&db..&ts..P&pr..&dsufb,
)ENDSEL
//        DCB=(BUFNO=2Ø,BLKSIZE=22528),
//        DISP=(,CATLG,DELETE),
//        SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)ENDDOT
)ENDSEL
//SYSIN   DD  *
)BLANK 1
)DOT "TLIST"
)SEL &scu GT Ø
  REORG    TABLESPACE &db..&ts
           LOG &log
)SEL &pr GT Ø
           PART &pr
)ENDSEL
)ENDSEL
)SEL &scu EQ Ø AND &obj EQ IX
```

```
  REORG    INDEX &cr..&ix
)SEL &pr GT Ø
           PART &pr
)ENDSEL
)ENDSEL
)SEL &sor = YES AND &scu GT Ø
           SORTDATA SORTKEYS
           SORTDEVT SYSDA SORTNUM 4
)ENDSEL
)SEL &sor = NO  AND &scu GT Ø
           SORTKEYS
           SORTDEVT SYSDA SORTNUM 4
)ENDSEL
)SEL &scu GT Ø
           SHRLEVEL &rtype
)SEL &rtype EQ REFERENCE OR &rtype EQ CHANGE
           COPYDDN COPY&scu
)ENDSEL
ENDSEL
)SEL &scu GT Ø AND &rtype EQ CHANGE
           MAPPINGTABLE SYSADM.MAPPTB
)ENDSEL
)SEL &dic = YES AND &scu GT Ø
           KEEPDICTIONARY
)ENDSEL
)ENDDOT
)SEL &rru = YES
)BLANK 1
)DOT "TLIST"
)SEL &scu GT Ø
  RUNSTATS TABLESPACE &db..&ts
)ENDSEL
)SEL &pr GT Ø AND &scu GT Ø
                 PART &pr
)ENDSEL
)SEL &scu EQ Ø AND &pr EQ Ø
  RUNSTATS INDEX ( &cr..&ix )
)ENDSEL
)SEL &scu EQ Ø AND &pr GT Ø
  RUNSTATS INDEX ( &cr..&ix PART &pr)
)ENDSEL
)ENDDOT
)ENDSEL
)SEL &ico = AFTER | &ico = BOTH
/*
//*── AFTER  IMAGE COPY  ─────────────────
//COPYA EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//      UID='&user..COPYA',UTPROC='',COND=(4,LT)
//STEPLIB DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
)DOT "TLIST"
)SEL &scu GT Ø
```

```
)SEL &ivol EQ Ø
//COPA&scu DD UNIT=SYSDA,
)ENDSEL
)SEL &ivol EQ 1
//COPA&scu DD UNIT=339Ø,VOL=SER=&vol,
)ENDSEL
)SEL &pr EQ Ø
//    DSN=&pref..&db..&ts..&dsufa,
)ENDSEL
)SEL &pr GT Ø
//    DSN=&pref..&db..&ts..P&pr..&dsufa,
)ENDSEL
//    DCB=(BUFNO=2Ø,BLKSIZE=22528),
//    DISP=(,CATLG,DELETE),
//    SPACE=(TRK,(&pri,&sec,),RLSE)
)ENDSEL
)ENDDOT
//SYSIN  DD  *
)DOT "TLIST"
)SEL &scu GT Ø
)BLANK 1
  COPY  TABLESPACE &db..&ts
)SEL &pr GT Ø
             DSNUM &pr
)ENDSEL
           COPYDDN COPA&scu
              FULL YES
          SHRLEVEL REFERENCE
)ENDSEL
)ENDDOT
)ENDSEL
/*
)SEL &rtype EQ NONE
//*── START TABLESPACES ────────────────
//STARTS  EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(&db2)
)BLANK 1
)DOT "TLIST"
)SEL &pr EQ Ø
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(RW)
)ENDSEL
)SEL &pr GT Ø
   -START DATABASE(&db) SPACENAM(&ts) ACCESS(RW) PART(&pr)
)ENDSEL
)ENDDOT
)ENDSEL
//*
)SEL &rtype EQ NONE
```

```
)SEL &obj EQ TS OR &obj EQ BOTH
)SEL &log EQ NO
)SEL &ico EQ NONE OR &ico EQ BEFORE
//*── COPY PENDING ──────────────────
//REPAIR  EXEC DSNUPROC,SYSTEM=&db2,COND=(4,LT)
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR,
//            DCB=BLKSIZE=32760
//SYSIN    DD *
)DOT "TLIST"
)SEL &scu GT 0
)SEL &pr EQ 0
  REPAIR SET TABLESPACE &db..&ts
             NOCOPYPEND
)ENDSEL
)SEL &pr GT 0
  REPAIR SET TABLESPACE &db..&ts
             NOCOPYPEND PART &pr
)ENDSEL
)ENDSEL
)ENDDOT
//*
)ENDSEL
)ENDSEL
)ENDSEL
)ENDSEL
```

Bernard Zver
Database Administrator
Informatika Maribor (Slovenia)

A real-time Coupling Facility monitor – part 1

When you decide to implement a Parallel Sysplex, you will need a tool to tune your Coupling Facility activity. You will have to check structure allocations in your different Coupling Facilities, structure response times, structure access rates, etc.

One solution is to use RMF to produce reports about Coupling Facilities and structures. The major drawback with this is that it is not a real-time process. To get real-time information about your Coupling Facility configuration, IBM provides a basic set of MVS commands to monitor your coupling facility configuration. To obtain the information you need, you should use the different forms of the D

XCF, etc command. For example, to get information about your DB2 lock structure, you would use the following command:

```
D XCF,STR,STRNAME=DB2M_LOCK1
```

To get the following result:

```
IXC360I  15.05.09  DISPLAY XCF 173
STRNAME: DB2M_LOCK1
 STATUS: ALLOCATED
  POLICY SIZE    : 49152 K
  POLICY INITSIZE: 32768 K
  REBUILD PERCENT: 1
  PREFERENCE LIST: ACF3     ACF1     L2A3XCF
  EXCLUSION LIST IS EMPTY

 ACTIVE STRUCTURE
 ----------------

  ALLOCATION TIME: 09/16/1999 17:40:51
  CFNAME         : ACF3
  COUPLING FACILITY: 009672.IBM.51.000000069150
                     PARTITION: 1   CPCID: 00
  ACTUAL SIZE    : 32768 K
  STORAGE INCREMENT SIZE: 256 K
  VERSION        : B11087A3 AEC8A301
  XCF GRPNAME    : IXCLO005
  DISPOSITION    : KEEP
  ACCESS TIME    : 0
  MAX CONNECTIONS: 7
  # CONNECTIONS  : 2

 CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
 ---------------- -- -------- -------- -------- ---- --------
 DXRM$$$$$FRLM002 02 00020022 FMVS     DB2FIRLM 009D ACTIVE
 DXRM$$$$$IRLM001 01 00010016 PROD     DB2AIRLM 0191 ACTIVE
```

Or, if you need to get information about the whole Coupling Facility, you will have to use the following command:

```
D XCF,CF,CFNAME=ACF3
```

To produce the following result:

```
IXC362I  15.06.32  DISPLAY XCF 515
CFNAME: ACF3
COUPLING FACILITY     :   009672.IBM.51.000000069150
                          PARTITION: 1   CPCID: 00
  POLICY DUMP SPACE SIZE:   10240 K
  ACTUAL DUMP SPACE SIZE:   10240 K
  STORAGE INCREMENT SIZE:   256 K
```

```
──────────────────────  Coupling Facility Monitor ──────────────────────────
 OPTION  ===>
                                                          USERID   - I990557
                                                          TIME     - 15:25

    C  Coupling Facility  - Coupling Facilities Display
    S  Structure          - CF Structures Display
```

Figure 1: Main panel

```
CONNECTED SYSTEMS:
FMVS     PROD

STRUCTURES:
DB2M_GBPØ         DB2M_GBP1         DB2M_GBP1Ø        DB2M_LOCK1
DB2M_SCA          IEFAUTOS          ISTGENERIC        IXC_PATH3
SYSTEM_LOGREC     SYSTEM_OPERLOG
```

These commands are not very user-friendly for systems programmers or operators. An additional drawback is that they do not provide any performance information about Coupling Facility activity.

```
──────────────  Coupling Facilities info ─────  Row 1 to 2 of 2
  COMMAND ===>                                          SCROLL ===> PAGE

 ──────────────────────────────────────
    CFNAME: ACF1
      Node: ØØ9674.IBM.51.ØØØØØØØ68216 PARTITION: Ø1 CPCID: ØØ
      CFLEVEL: 4           Storage Increment:     256 k   Volatile: N
      Storage Usage:
        Total:  119808  k
        Dump :   10240  k
        Free :  105472  k
 ──────────────────────────────────────
    CFNAME: ACF3
      Node: ØØ9672.IBM.51.ØØØØØØØ6915Ø PARTITION: Ø1 CPCID: ØØ
      CFLEVEL: 6           Storage Increment:     256 k   Volatile: Y
      Storage Usage:
        Total:  250368  k
        Dump :   10240  k
        Free :  141568  k
```

Figure 2: Coupling facilities panel

Another approach is to use a third-party product (Boole & Babbage, Candle, etc). However, these products do not always provide all the information you need and want. This is why I decided to write my own utility to monitor Coupling Facilities and structures, implemented as an ISPF application.

The main panel gives you two choices, as shown in Figure 1. The first choice (C) allows you to list Coupling Facilities connected to your MVS system, as shown in Figure 2. This panel gives you information on storage usage of your Coupling Facilities, micro code CFLEVEL, and volatility status. Then, you can select one of the Coupling Facilities to get the list of its allocated structures. This is shown in Figure 3.

At that time, you get real-time statistics on structure storage usage. When you hit enter, the panel is automatically refreshed. You can also select one structure to get more detailed information (see Figure 4).

```
──────────────────────────── Coupling Facility Detail ──────── Row 1 to 7 of 10
COMMAND ===>                                                  SCROLL ===> PAGE

CFNAME: ACF3
    Dump:   10240 k Structures:   98560 k Free:  141568 k Total:  250368 k

  STRNAME          Storage Type    Lst/Dir Lst     Data    Lock    Lock/Elm
                     (k)           Entries Headers Element Entries Size (b)
                                   Tot/Use         Tot/Use
  DB2M_GBP0          32768 CACHE     32244             6449             4096
                                      3082             3082
  DB2M_GBP1           8192 CACHE      7987             1594             4096
                                         3                3
  DB2M_GBP10          2048 CACHE      1908              378             4096
                                         0                0
  DB2M_LOCK1         32768 LOCK                                8388608        2
                                                                 88149
  DB2M_SCA           16384 LIST      24840     200     49679              256
                                        89              179
  IEFAUTOS            1024 LIST       1984     266      1984              256
                                         0                0
  ISTGENERIC           256 LIST        884       4        15             1024
                                        14                3
```

Figure 3: Coupling Facility detail panel

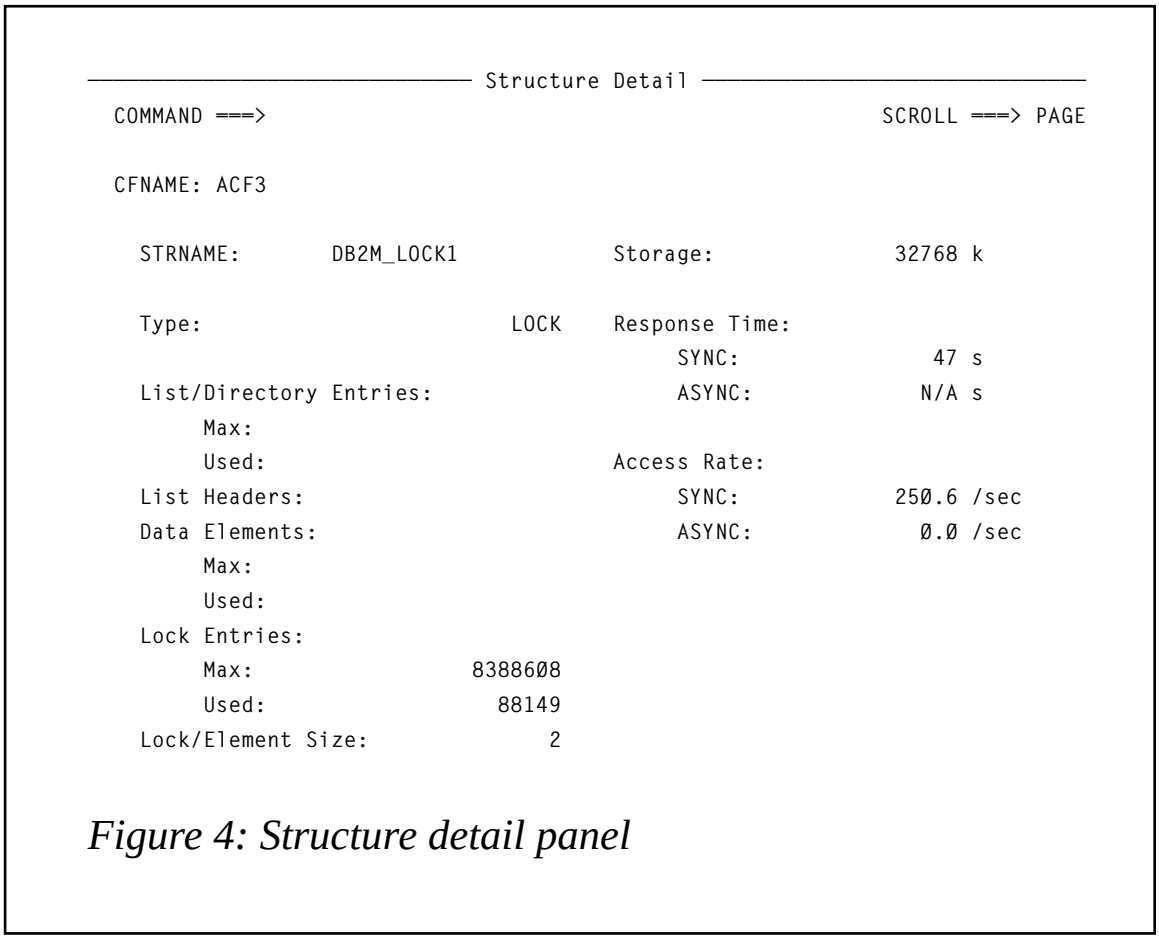

Figure 4: Structure detail panel

You obtain response time and access rate information on the selected structure. The figures are also refreshed when you hit enter.

STRUCTURE PANELS

The second choice (S) of the main panel allows you to list structures connected to your MVS system (see Figure 5).

This panel provides structure definition data specified in active CFRM policy. You can select one structure to get information about active connections (see Figure 6). This panel provides information about the ‘ability’ to alter and rebuild the structure.

IMPLEMENTATION

The monitor is designed around two Assembler programs (IXCCFIS and IXCSTIS).These two programs use two Sysplex services macros

```
──────────────────────────── Structures Info ──────────────  Row 1 to 3 of 11
COMMAND ===>                                                 SCROLL ===> PAGE
──────────────────────────────────────────────────────────────────────────────

  STRNAME: DB2M_GBP0         Status: ALLOCATED
       CF: ACF3         00009672.IBM.51.000000006915Ø PARTITION: Ø1 CPCID: ØØ
       Preference List: ACF3     ACF1     L2A3XCF
       Exclusion List: LIST IS EMPTY
       Initsize:   32768 K - Size:   49152 K - Rebuild Pct:       1
──────────────────────────────────────────────────────────────────────────────

  STRNAME: DB2M_GBP1         Status: ALLOCATED
       CF: ACF3         00009672.IBM.51.000000006915Ø PARTITION: Ø1 CPCID: ØØ
       Preference List: ACF3     ACF1     L2A3XCF
       Exclusion List: LIST IS EMPTY
       Initsize:    8192 K - Size:   12288 K - Rebuild Pct:       1
──────────────────────────────────────────────────────────────────────────────

  STRNAME: DB2M_GBP10        Status: ALLOCATED
       CF: ACF3         00009672.IBM.51.000000006915Ø PARTITION: Ø1 CPCID: ØØ
       Preference List: ACF3     ACF1     L2A3XCF
       Exclusion List: LIST IS EMPTY
       Initsize:    2048 K - Size:    4096 K - Rebuild Pct:       1
```

Figure 5: Structures panel

– IXLMG and IXCQUERY. You can get detailed information about these macros in *OS/390 MVS Programming: Sysplex Services Reference* (GC28-1772).

```
──────────────────────────── Structures Info ──────────────── Row 1 to 2 of 2
COMMAND ===>                                                 SCROOL ===> PAGE

  STRNAME: DB2M_GBP0         STATUS: ALLOCATED
       CF: ACF3         00009672.IBM.51.000000006915Ø PARTITION: Ø1 CPCID: ØØ
       Prefrence List: ACF3     ACF1     L2A3XCF
       Exclusion List: LIST IS EMPTY
       Initsize:   32768 k - Size:   49152 k - Rebuild pct:       1

                                                        Allow
Connection Name  Sysname  Jobname  Status           Alter    Rebuild
──────────────────────────────────────────────────────────────────────────────
DB2_DB2A         PROD     DB2ADBM1 ACTIVE           Y        Y
DB2_DB2F         FMVS     DB2FDBM1 ACTIVE           Y        Y
****************************** BOTTOM OF DATA ********************************
```

Figure 6: Structure detail panel

IXCCFIS

IXCCFIS uses the IXLMG macro to retrieve information about Coupling Facilities connected to your MVS system. The IXLMG macro allows an authorized caller to request measurement data related to the use of a Coupling Facility.

Structures in the IXLYAMDA mapping macro provide the format for the returned data:

- IXLYAMDAREA maps the header record.
- XLYAMDCF maps Coupling Facility information.
- IXLYAMDSTRL maps Coupling Facility list structure information.
- IXLYAMDSTRC maps Coupling Facility cache structure information.

SOURCE CODE

```
IXCCFIS  CSECTIXCCFIS
         AMODE 31IXCCFIS
         RMODE ANY*
         USING IXCCFIS,R15*
         SAVE  (14,12)
         LR    R12,R15
         DROP  R15
*         LA    R11,2Ø48(R12)
         LA    R11,2Ø48(R11)
*         USING IXCCFIS,R12,R11
*         GETMAIN R,LV=WORKL
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
*         USING DSECT,R13
*
* CREATE ISPF VARIABLES
         CALL  ISPLINK,(VDEFINE,FZTDSELS,ZTDSELS,CHAR,L4),VL
         CALL  ISPLINK,(VDEFINE,FSELECT,SELECT,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FCFNAME,CFNAME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FCFNODE,CFNODE,CHAR,L54),VL
         CALL  ISPLINK,(VDEFINE,FTSPACE,TSPACE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FFSPACE,FSPACE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FDSPACE,DSPACE,CHAR,L8),VL
```

```
         CALL  ISPLINK,(VDEFINE,FCFLEVEL,CFLEVEL,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTINC,STINC,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FVOL,VOL,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FSTRNAME,STRNAME,CHAR,L16),VL
         CALL  ISPLINK,(VDEFINE,FSTSIZE,STSIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTTOT,STTOT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTTYPE,STTYPE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLENT,STLENT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLENTU,STLENTU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDELM,STDELM,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDELMU,STDELMU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLSIZE,STLSIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLHD,STLHD,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDENT,STDENT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDENTU,STDENTU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTSTIME,STSTIME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTATIME,STATIME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTSCNT,STSCNT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTACNT,STACNT,CHAR,L8),VL
*
*        CREATE AND SORT ISPF TABLES
* CREATE ISPF TABLE
         CALL ISPLINK,(TBCREATE,FSTABLE,,NAMELIST,NOWRITE,REPLACE),VL
         MVC  SORTKEY,FCFNAME                    DEFAULT SORT
         MVC  SORTTYPE,CHARASND
*        CALL ISPLINK,(TBSORT,FSTABLE,SORTPARM),VL
*
* PROGRAM CODE
         LA    R9,ANSAREA
         USING IXLYAMDAREA,R9
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT…
         LTR   R1Ø,R1Ø
         BZ    RETURN
         USING IXLYAMDCF,R1Ø
*
* CF PANEL
*
LOOPCF   EQU   *
*                                                SET DEFAULT VALUES
         MVC   CFNAME,IXLYAMDCF_CFNAME
         MVC   VOL,=CL1"Y'
         TM    IXLYAMDCF_FLAGS,IXLYAMDCF_VOLATILE
         BNZ   FLAGV
         MVC   VOL,=CL1"N'
FLAGV    EQU   *
         LA    R7,IXLYAMDCF_ND
         USING NDE,R7
         MVC   CFNODE(6),NDETYPE
```

```
         MVI   CFNODE+6,C'.'
         MVC   CFNODE+7(3),NDEMFG
         MVI   CFNODE+1Ø,C'.'
         MVC   CFNODE+11(2),NDEPLANT
         MVI   CFNODE+13,C'.'
         MVC   CFNODE+14(12),NDESEQUENCE
         MVC   CFNODE+27(1Ø),=C'PARTITION:'
         XR    R8,R8
         IC    R8,NDEPARTITION               * PARTITION
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+38(2),DOUBLE+1
         MVC   CFNODE+41(6),=C'CPCID:'
         XR    R8,R8
         IC    R8,NDECPCID                   * CPCID
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+48(2),DOUBLE+1
         DROP  R7
         L     R8,IXLYAMDCF_TS               TOTAL SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   TSPACE,MASK
         ED    TSPACE(8),DOUBLE+4
         L     R8,IXLYAMDCF_FS               FREE  SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   FSPACE,MASK
         ED    FSPACE(8),DOUBLE+4
         SR    R8,R8
         LH    R8,IXLYAMDCF_STGI             STORAGE INCREMENT
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   STINC,MASK
         ED    STINC(8),DOUBLE+4
         L     R8,IXLYAMDCF_TDS              DUMP  SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   DSPACE,MASK
         ED    DSPACE(8),DOUBLE+4
         L     R8,IXLYAMDCF_CFLEVEL          CFLEVEL
         CVD   R8,DOUBLE
         MVC   CFLEVEL,MASK
         ED    CFLEVEL(8),DOUBLE+4
*
* ADD A NEW ROW
         CALL  ISPLINK,(TBADD,FSTABLE),VL  ADD DATA INTO TABLE
```

```
         L     R1Ø,IXLYAMDCF_CFNEXT
         LTR   R1Ø,R1Ø                          LAST CF ENTRY ?
         BNZ   LOOPCF
DISPP1   EQU   *
         CALL  ISPLINK,(TBSORT,FSTABLE,SORTPARM),VL  SORT TABLE
         CALL  ISPLINK,(TBTOP,FSTABLE),VL  POINT TO TOP OF TABLE
         CALL  ISPLINK,(TBDISPL,FSTABLE,FSPANEL),VL  DISPLAY TABLE
         C     R15,=F'8'               HAS PF3   BEEN HIT (R15 = 8)?
         BE    RETURN                  NO, TERMINATE PROGRAM
         CLC   SELECT,=CL1"S'
         BNE   NOSELECT
*
* CF / STR PANEL
*
FLAG1Ø   EQU   *
         CALL ISPLINK,(TBCREATE,FSTABLES,,NAMELISS,NOWRITE,REPLACE),VL
         MVC  SORTKEYS,FSTRNAME                  DEFAULT SORT
         MVC  SORTTYPS,CHARASND
*         CALL ISPLINK,(TBSORT,FSTABLE,SORTPARS),VL
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT…
         USING IXLYAMDCF,R1Ø
LOOPCFS  EQU   *
         CLC   CFNAME,IXLYAMDCF_CFNAME
         BE    GOTCF
         L     R1Ø,IXLYAMDCF_CFNEXT
         B     LOOPCFS
GOTCF    EQU   *
         SR    R5,R5                        TOTAL STRUCTURES SIZE
*
         L     R4,IXLYAMDCF_STR…
         LTR   R4,R4
         BZ    FLAGØ2
LOOPSTR  EQU   *
         BAL   R6,GETSTR
         USING IXLYAMDSTRL,R4
         CALL  ISPLINK,(TBADD,FSTABLES),VL  ADD DATA INTO TABLE
         L     R4,IXLYAMDSTRL_STRNEXT
         LTR   R4,R4                        LAST ENTRY ?
         BNZ   LOOPSTR
FLAGØ2   EQU   *
         CVD   R5,DOUBLE
         MVC   STTOT,MASK
         ED    STTOT(8),DOUBLE+4
*
         CALL  ISPLINK,(TBSORT,FSTABLES,SORTPARS),VL  SORT TABLE
         CALL  ISPLINK,(TBTOP,FSTABLES),VL  POINT TO TOP OF TABLE
REDISPP2 CALL  ISPLINK,(TBDISPL,FSTABLES,FSPANELS),VL  DISPLAY TABLE
*
```

```
         C     R15,=F'8'               HAS PF3   BEEN HIT (R15 = 8)?
         BE    DISPP1                  NO, TERMINATE PROGRAM
         MVC   OTIME,=2D'Ø'
         MVC   OSTIMEC,=F'Ø'
         MVC   OSTIMES,=D'Ø'
         MVC   OATIMEC,=F'Ø'
         MVC   OATIMES,=D'Ø'
         MVC   STSTIME,=CL8"N/A'
         MVC   STATIME,=CL8"N/A'
         MVC   STSCNT,=CL8"N/A'
         MVC   STACNT,=CL8"N/A'
         CLC   SELECT,=CL1"S'
         BE    REDISPP3
         CALL  ISPLINK,(TBCLOSE,FSTABLES),VL  CLOSE TABLE
         B     FLAG1Ø
*
* STR DETAIL PANEL
*
REDISPP3 EQU   *
         STCKSYNC TOD=NTIME            GET REQUEST TIME
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT…
         USING IXLYAMDCF,R1Ø
LOOPCFSD EQU   *
*
         CLC   CFNAME,IXLYAMDCF_CFNAME
         BE    GOTCFD
         L     R1Ø,IXLYAMDCF_CFNEXT
         B     LOOPCFSD
GOTCFD   EQU   *
         L     R4,IXLYAMDCF_STR…
         LTR   R4,R4
         BZ    FLAGØ2
LOOPSTRD EQU   *
         USING IXLYAMDSTRL,R4
         CLI   IXLYAMDSTRL_TYPE,X'21'
         BNE   STCACHED                        * CACHE STRUCTURE ?
         CLC   STRNAME,IXLYAMDSTRL_STRNAME
         BE    GOTSTD
         B     GETSTDE
```

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

Patrick Renard
CTRNE (France)

Improving DISTINCT's performance of SQL in DB2

DISTINCT is a feature of SQL that is frequently used by programmers to eliminate the occurrence of duplicates in the answer set. If a column on which DISTINCT needs to be applied is one of the index columns, DB2 should do an Indexspace scan to complete this query. However, if the percentage of rows in the table (filter factor) with unique values in the column on which DISTINCT is to be applied is relatively small, then DB2 might not use indexspace scan. Even if DB2 uses indexspace scan, it might still go for a sort on the composite table to achieve the objective. This is very costly for a big table, for example with 686,036 rows with 6.33% unique rows for a column on which DISTINCT needs to be applied.

This article offers an alternative that will prevent DB2 carrying out a sort on the composite table. It analyses information about the various access paths used by DB2 to obtain the desired objective, and the DB2 CPU time taken for various cases.

The benchmarking results are very encouraging. By using the new method, DB2 CPU usage is seen to improve 46,200 times, on a table (with 686,036 rows) having 6.33% unique rows for that column. It is also seen to improve 262,402 times if the table has 4.4 million rows, with 59% unique rows for the column on which DISTINCT is applied.

This recommendation is benchmarked with DB2 CPU usage and plan_table statistics to ascertain the fact that DISTINCT's performance can be vastly improved by using GROUP BY.

INTRODUCTION

We use DISTINCT to eliminate the occurrence of duplicates for a column in our answer set. It is felt that, when DISTINCT is used on a primary key column, DB2 should always use Indexspace scan. The decision by DB2 to use Indexspace scan is influenced by the percentage (filter factor) of unique rows, for those columns used in the DISTINCT clause. If the percentage of unique rows is low, the cost of sorting a

composite table is relatively high. Hence, sorting a big composite table by doing a composite tablespace scan poses a big threat to performance.

ANALYSIS

Let us analyse the following four SQL examples. The first two SQL examples (namely SQL1 and SQL2) attempt to use DISTINCT on the first column of the index. The next two SQL examples (SQL3 and SQL4) attempt to use DISTINCT on the first two columns of the index.

SQL1:

```
DECLARE CUSMIN_CSR CURSOR FOR
  SELECT DISTINCT ( AC_NR )
  FROM TCUSMIN
  ORDER BY AC_NR
  FOR FETCH ONLY ;
```

where

- C1 is the composite table on which DB2 performs sorting to 'check for uniqueness' and 'order by'.
- Accesstype 'R' indicates that DB2 is doing a tablespace scan for SQL1.

TCUSMIN is a table with 686,036 rows with 43,435 DISTINCT AC_NR (6.33% of rows in TCUSMIN have unique AC_NR). AC_NR is the leading primary key of the table tcusmin. When the cursor on SQL1 is opened, it took 6.93 CPU seconds to process the SQL, which is very high for any SQL. The Platinum detector information and plan table analysis are shown in Figure 1.

SQL2:

```
DECLARE ACCADR_CSR CURSOR FOR
  SELECT DISTINCT ( AC_NR )
  FROM TACCADR
  ORDER BY AC_NR
  FOR FETCH ONLY ;
```

TACCADR is a table with 4,341,727 rows with 2,590,864 DISTINCT

SQL1

```
QBLOCK# PLANNO  MTH  TNAME    TABNO ACCESS TYPE  MATCH COLS ACCESS NAME  INDEX ONLY  SORTC_UNIQ SORTC_ORDERBY
1       1       Ø    TCUSMIN  1     R            ØØ                    N           N          N
1       2       3    C1       Ø                  ØØ                    N           Y          Y
   CALL          # OF SQLS      TIMEPCT                  CPUPCT                    INDB2_TIME    INDB2_CPU
   OPEN          1              99.99%                   99.99%                    1Ø.282877SEC  6.939210SEC
```

SQL2

```
QBLOCK# PLANNO  MTH  TNAME    TAB#  ACCESS TYPE  MATCH COLS ACCESS NAME  INDEX ONLY  SORTC_UNIQ SORTC_ORDERBY
1       1       Ø    TACCADR  1     I            ØØ        IACCADR1    Y           N          N
1       2       3    C2       Ø                  ØØ                    N           Y          Y
   CALL          # OF SQLS      TIMEPCT                  CPUPCT                    INDB2_TIME    INDB2_CPU
   OPEN          1              99.99%                   99.99%                    85.012132SEC  36.474303SEC
```

SQL3

```
QBLOCK# PLANNO  MTH  TNAME    TAB#  ACCESS TYPE  MATCH COLS ACCESS NAME  INDEX ONLY  SORTC_UNIQ SORTC_ORDERBY
1       1       Ø    TCUSMIN  1     R            ØØ                    N           N          N
1       2       3    C3       Ø                  ØØ                    N           Y          Y
   CALL          # OF SQLS      TIMEPCT                  CPUPCT                    INDB2_TIME    INDB2_CPU
   OPEN          1              99.99%                   99.99%                    13.850743SEC  7.661208SEC
```

SQL4

```
QBLOCK# PLANNO  MTH  TNAME    TAB#  ACCESS TYPE  MATCH COLS ACCESS NAME  INDEX ONLY  SORTC_UNIQ SORTC_ORDERBY
1       1       Ø    TACCADR  1     I            ØØ        IACCADR1    Y           N          N
1       2       3    C4       Ø                  ØØ                    N           Y          Y
   CALL          # OF SQLS      TIMEPCT                  CPUPCT                    INDB2_TIME    INDB2_CPU
   OPEN          1              99.99%                   99.99%                    87.040004SEC  38.924077SEC
```

Figure 1: Results

AC_NR (59.6% of rows in TACCADR have unique AC_NR). AC_NR is the leading primary key of the table TACCADR. When the cursor on SQL2 is opened, it takes 36.47 CPU seconds to process the SQL. The Platinum detector information is shown in Figure 1. Even if DB2 performs an Indexspace scan through index IACCADR1, because of a high percentage of unique rows, it still does a sort on composite table C2 for 'check for uniqueness' and 'order by'.

We now attempt to use DISTINCT on the first two columns of the index of tcusmin and taccadr in SQL3 and SQL4 respectively.

SQL3:

```
DECLARE CUSMIN_CSR CURSOR FOR
  SELECT DISTINCT AC_NR, CNY_CD
  FROM TCUSMIN
  ORDER BY AC_NR, CNY_CD
  FOR FETCH ONLY ;
```

TCUSMIN is a table with 686,036 rows with 43,435 DISTINCT (AC_NR, CNY_CD) (6.33% of rows in TCUSMIN are unique for AC_NR and CNY_CD). AC_NR and CNY_CD are the two leading columns of the index of the table tcusmin. When the cursor on SQL3 is opened, it took 7.66 CPU seconds to process the SQL. The Platinum detector information is shown in Figure 1.

Even if we try to get more columns of the index into our DISTINCT clause, the performance remains unchanged. DB2 does a tablespace scan in addition to sorting the composite table C3 to 'check for uniqueness' and 'order by'.

SQL4:

```
DECLARE ACCADR_CSR CURSOR FOR
  SELECT DISTINCT AC_NR, AC_TYP_CD
  FROM TACCADR
  ORDER BY AC_NR, AC_TYP_CD
  FOR FETCH ONLY ;
```

TACCADR is a table with 4,341,727 rows with 2,590,864 DISTINCT AC_NR and AC_TYP_CD (59.6% of rows in TACCADR for these two columns are unique). AC_NR and AC_TYP_CD are the leading primary keys of the table TACCADR. When the cursor on SQL4 is opened, it takes 38.92 CPU seconds to process the SQL. The Platinum

detector information is shown in Figure 1.

Even if DB2 uses Indexspace scan on index IACCADR1 because of the high percentage of unique rows, it still does a sort on composite table C4 to 'check for uniqueness' and 'order by'.

Now we have seen that DISTINCT does not use Indexspace scan for a percentage of unique rows of 6.33%. However, on a table with a high percentage of unique rows (59.6%), even if it does an Indexspace scan, it does a composite table sort to determine uniqueness and order by.

Now let us rewrite the query by using 'GROUP BY' without specifying any statistical functions (like AVG, MAX, and MIN).

SQL5 gives the same result as SQL1.

SQL6 gives the same result as SQL2.

SQL7 gives the same result as SQL3.

SQL8 gives the same result as SQL4.

We try to use the GROUP BY feature, which is used for statistical functions, to exploit DISTINCT's non-performance. Now, let us analyse the results.

SQL5:

```
DECLARE CUSMIN_CSR CURSOR FOR
  SELECT AC_NR
  FROM TCUSMIN
  GROUP BY AC_NR
  ORDER BY AC_NR
  FOR FETCH ONLY ;
```

When the cursor on SQL5 is opened, it takes 0.150 CPU seconds to process the SQL – very low for any SQL, and 46,200 times faster than SQL1. The Platinum detector information and plan table analysis are shown in Figure 2. It uses index ICUSMIN1 and does not do a sort on the composite table, in contrast to what DB2 was doing in SQL1.

SQL6:

```
DECLARE ACCADR_CSR CURSOR FOR
```

SQL5

```
QBLOCK#  PLANNO  MTH  TNAME     TAB#  ACCESS TYPE  MATCH COLS  ACCESS NAME  INDEX ONLY  SORTC_UNIQ SORTC_ORDERBY
1        1       Ø    TCUSMIN   1     I            ØØ          ICUSMIN1     Y           N
   CALL                # OF SQLS      TIMEPCT                  CPUPCT                   INDB2_TIME  INDB2_CPU
   OPEN                1              85.78%                   84.26%                   0.169 ms    0.150 ms
```

SQL6

```
QBLOCK#  PLANNO  MTH  TNAME     TAB#  ACCESS TYPE  MATCH COLS  ACCESS NAME  INDEX ONLY  SORTC_UNIQ  SORTC_ORDERBY
1        1       Ø    TACCADR   1     I            ØØ          IACCADR1     Y           N           N
   CALL                # OF SQLS      TIMEPCT                  CPUPCT                   INDB2_TIME  INDB2_CPU
   OPEN                1              82.73%                   82.73%                   0.139 ms    0.139 ms
```

SQL7

```
QBLOCK#  PLANNO  MTH  TNAME     TAB#  ACCESS TYPE  MATCH COLS  ACCESS NAME  INDEX ONLY  SORTC_UNIQ  SORTC_ORDERBY
1        1       Ø    TCUSMIN   1     I            ØØ          ICUSMIN1     Y           N           N
   CALL                # OF SQLS      TIMEPCT                  CPUPCT                   INDB2_TIME  INDB2_CPU
   OPEN                1              84.84%                   84.84%                   0.140 ms    0.140 ms
```

SQL8

```
QBLOCK#  PLANNO  MTH  TNAME     TAB#  ACCESS TYPE  MATCH COLS  ACCESS NAME  INDEX ONLY  SORTC_UNIQ  SORTC_GROUPBY
1        1       Ø    TACCADR   1     I            ØØ          IACCADR1     Y           N           N
1        2       3    C8        Ø                  ØØ                       N           N           Y
   CALL                # OF SQLS      TIMEPCT                  CPUPCT                   INDB2_TIME  INDB2_CPU
   OPEN                1              99.99%                   99.99%                   54.912809 sec  35.554010 sec
```

Figure 2: Results

```
  SELECT AC_NR
  FROM TACCADR
  GROUP BY AC_NR
  ORDER BY AC_NR
  FOR FETCH ONLY ;
```

When the cursor on SQL6 is opened, it takes 0.139 CPU seconds to process the SQL, which is 262,402 times faster as far as the CPU is concerned. It uses index IACCADR1 and does not do a sort on the composite table, in contrast to what DB2 was doing in SQL2. The results are shown in Figure 2.

SQL7:

```
DECLARE CUSMIN_CSR CURSOR FOR
  SELECT AC_NR, CNY_CD
  FROM TCUSMIN
  GROUP BY AC_NR, CNY_CD
  ORDER BY AC_NR, CNY_CD
  FOR FETCH ONLY ;
```

When the cursor on SQL7 is opened, it takes 0.140 CPU seconds to process the SQL, which is 54,721 times faster than SQL3 as far as the CPU is concerned. The Platinum detector information is shown in Figure 2.

SQL8:

```
DECLARE ACCADR_CSR CURSOR FOR
  SELECT AC_NR, AC_TYP_CD
  FROM TACCADR
  GROUP BY AC_NR, AC_TYP_CD
  ORDER BY AC_NR, AC_TYP_CD
  FOR FETCH ONLY ;
```

When the cursor on SQL8 is opened, it takes 35.55 CPU seconds to process the SQL, which is 1.09 times faster than SQL4 as far as the CPU is concerned. The extent of CPU utilization in SQL8 is comparatively high compared with SQL5, SQL6, and SQL7 because DB2 is trying to do a sort on a composite table for ‘GROUP BY’, unlike SQL5, SQL6, and SQL7. The Platinum detector information is shown in Figure 2.

CONCLUSIONS AND RECOMMENDATIONS

DB2 'GROUP BY' groups data by using statistical functions on a column. It applies statistical functions to consolidate the data. GROUP BY does not always sort the data to arrive at the result set, as we have seen in the plan table information.

When we don't provide any statistical function (MAX, MIN, AVG etc) to GROUP BY, DB2 performs much better, because it doesn't need to group data for statistical computations. Secondly, it does not need to do a sort on a composite table to 'check for uniqueness'. If the GROUP BY columns are part of the index, then DB2 will probably (depending on the filter factor, of course!) use indexspace scan on the index columns. While using DISTINCT, as we have seen, we are letting DB2 decide whether to use indexspace scan, by determining the percentage of unique rows for the index column. This can be avoided, if users find cardinality parameters from the catalog table earlier and use GROUP BY without any statistical function. GROUP BY without any statistical function in all cases will be faster than DISTINCT.

The recommendation is based on syntactically rearranging queries using SQL algebra:

```
Select distinct colm1, colm2, .., colmN from table1 order by colm1,
coml2, .., colmN
```

gives the same result as:

```
Select colm1, colm2, .., colmN from table1 group by by colm1, coml2, ..,
colmN
order by colm1, coml2, .., colmN
```

Hemanta Ranjan Panda
Computer Consultant
PricewaterhouseCoopers (USA)

Compuware has announced plans to support DB2 Universal Database with its QALoad performance and load testing product. Its latest version of QALoad now supports testing CORBA objects, including Iona's Orbix, Inprise's VisiBroker, and IBM's WebSphere.

QALoad supports the ability to perform load testing of applications using CORBA by recording the object calls a client application makes and converting these recordings into a test script.

Using this script, companies can simulate up to tens of thousands of actual users applying appropriate client load against the application infrastructure. Teams can then measure the end-to-end client response time under increasing load conditions.

Compuware's EasyScript for CORBA is available for Java applications and applets, which use either Inprise VisiBroker or Orbix.

For further information contact:
Compuware, 31440 Northwestern Highway, Farmington Hills, MI 48334-2564, USA.
Tel: (248) 737 7300.
Compuware, 163 Bath Road, Slough, SL1 4AA, UK.
Tel: (01753) 774000.
URL: http://www.compuware.com.

* * *

Merant has announced new DataDirect Connect ODBC drivers that support database specific protocols including DRDA for DB2 and TDS for Sybase and SQL Server. These allow direct connectivity to databases without requiring additional database vendor middleware or gateways.

DataDirect Connect Premium for DB2 offers direct access to DB2 using the DRDA protocol and includes support for Static SQL and other security and performance features.

For further information contact:
Merant, Speen Court, 7 Oxford Road, Newbury, Berks, RG14 1PB, UK.
Tel: (01635) 32646.
Merant, 2465 E Bayshore Road, Palo Alto, CA 94303, USA.
Tel: (650) 856 4161.
URL: http://www.merant.com.

* * *

IBM has announced Version 6.1 of its Enterprise Information Portal for the retrieval, dissemination, and use of available information.

Providing the foundation for a single point of access to both structured and unstructured information, regardless of origin, the new version supports federated search across the EDMSuite and DB2 Digital Library family of content servers.

Also, SAP and IBM have an expanded sales, marketing, and development relationship through which the two will work on providing DB2 on a variety of hardware platforms for mySAP.com Internet business solutions – including Windows 2000, AIX, OS/390, OS/400, Solaris, Linux, and NT.

DB2 Universal Database will replace Oracle as the primary database on IBM, Sun, and Linux platforms for internal development and production systems within SAP.

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