IDMS Connections

THE publication of the IDMS User Association

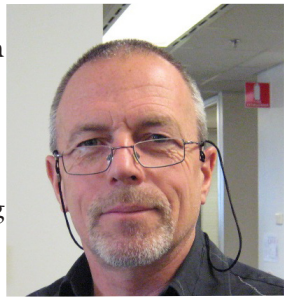
The CA-IDMS Database and Applications User Association

http://www.iuassn.com

September 2009, Number 71

INSIDE THIS ISSUE

LETTER FROM THE EDITOR



Can you believe it? September 2009. Children in North America are returning to school or just starting their first year of a life long learning process. As October nears our Southern Hemisphere children are readying themselves for school break and final exams. Then for all of us it's Christmas, New Year 2010 and time to pull together the March 2010 IUA Connections! Whew!

On the topic of time - how about this little gem of an exchange on IDMS-L?

From: IDMS 3rd-party providers forum on behalf of Chris Hoelscher
Sent: Thu 4/16/2009 12:01 PM
To: IDMSVENDOR-L@LISTSERV.IUASSN.COM
Subject: Happy Birthday?

From what I recall - 2009 is the 40th birthday of IDMS and the 35th birthday of IUA!! Happy Birthday!

From: IDMS Public Discussion Forum [IDMS-L@LISTSERV.IUASSN.COM] on behalf of Bob Wiklund [wiklund@TIBURONTECH.COM]
Sent: Friday, 17 April 2009 5:52
To: IDMS-L@LISTSERV.IUASSN.COM
Subject: Re: Happy Birthday?

Chris,

Per Peter Karasz' origins of IDMS, they started writing the DBMS in 1969 and finished in 1971. The IUA was incorporated March 6th, 1981. The original board of directors were: Van Banks of Harris Corp., George Bentley of AAI Corp., William Corward of Pullman Kellog, James Emerson of Peat, Marwick & Mitchell, Jerry McColough of Owens-Corning Fiberglass, John

(continued on page 3)

MESSAGE FROM INTERNATIONAL CHAIR

By Terry Schwartz

On September 15th the European IUA chair Jan Rabaut and I met with members of the CA IDMS management team. At this meeting we were in formed that Judy Kruntorad is taking a new assignment and will be stepping down as product owner of CA IDMS. Judy has been the product owner of CA IDMS for 20 years and a great deal has transpired in those 20 years.

If my history is correct, during the 20 years as product owner Judy presided over:

IDMS Release 12.0	- full SQL / IDMS Server
IDMS Release 14.0	- Multitasking
IDMS Release 14.1	- Multiple Page Groups
IDMS Release 15	- Data Sharing
IDMS Release 16	- TCP/IP
IDMS Release 17	– zIPP Support

Since I have been involved in the IUA (1997) Judy has demonstrated unwavering support for CA IDMS user groups. In Judy I feel we have had a true partner who realized that we both have the IDMS users best interest at heart.

Art Cartier is taking over as the new product owner for IDMS. Art has been a product owner for CA testing tools and has been with CA for 18 years.

Art has a pre CA background with IMS but he said it was really old. (He has already been ribbed)

Judy will be taking on a new position as Northeastern region quality manager for a host of other products. We certainly look forward to working with Art and wish both Art and Judy the best in their new jobs.

WWW.IUASSN.ORG

YOUR PORTAL TO IUA SERVICES AND IDMS CONTACTS

Letter from the Editor cont'd from page 1

Peck of Clemson University, Frank Sweet of Chapter Computer Co., Larry Towner of RCA

Does anyone have any additional history of the IUA they would be willing to Share?

Bob Wiklund
IUA Board
Tiburon Technologies
623 594-6022

In 1969 I was at the Toronto Pop Festival - missed Woodstock a couple of weeks later, but then I was taking an upper air weather observing course at the time so it was difficult to get away. In 1981 I was making the big move from Canada to Australia, and yes (to quote a famous Australian singer-song writer), "I still call Australia home!". Where were you back then? How many of our readers or colleagues were even born then?

Enough about time and how it seems to fly by - on to this issue. Dare I say yet again, "Boy - do we have another bumper issue for you?". We have articles that should interest DBA's and application developers alike. Chris Hoelscher provides insight into zIP processors and the Mainframe Software Management (MSM) way in which software upgrades take place under Mainframe 2.0 - in this case starting with Release 17.0.

Kay Rozeboom provides the next in a series of excellent articles on XML - which I must point out is a technology that helps us to achieve the long sought holy grail of "data and vendor independence". And CA has an excellent article on the mysteries of some of the internals of data management inside IDMS databases.

I hope you enjoy this edition of IUA *Connections* - and that some of you will feel inspired to tell us some of your war stories accumulated at some stage during the last 40 years or so.

As for thanking those who have contributed to this and past issues - I think I will quote myself by thanking "our contributors from CA and our regular and ad-hoc contributors from within the IDMS User Community without whom producing this publication would not be possible. From myself, and on behalf of all the readers who gain from your stories and your insights into a wide range of topics - THANK YOU!".

That's all there is because there is no more – cheers - Gary

Gary Cherlet
Justice Technology Services
President Australian IDMS User Group (OZIUA)
IUA International Board Member responsible for *Connections*

The opinions expressed in this editorial are the personal opinions of the editor and they may not be shared by the IUA Board or its members, other contributors to Connections, by Justice Technology Services, or the Government of South Australia.

DEMI/DB – A USER EXPERIENCE AT LOCKHEED MARTIN

by James G Grobaker

Introduction

As a programmer/analyst at Lockheed Martin Corporation (LMC), Enterprise Business Services, for about 20 years (8 as an employee, and the prior 12 as a contractor), I have often needed to write stand-alone "one-shot" programs to extract, modify or import data into our production systems. Our CA-IDMS production systems include:

- Consolidated Purchasing System (CPS) – homegrown, originally based on PIOS
- PIOS-MRP – heavily modified version of M&D PIOS
- Customer Organization and Tracking System (CORT) – homegrown
- Material Estimating and Tracking System (METS) – homegrown

I primarily work on CPS, METS, and a system called Buyer's Work Station (BWS), which is an Oracle add-on system to CPS. As I mentioned, I have had to write "one-shot" programs many times in the past, but that was before LMC purchased the Data Extract Modification and Import (DEMI/DB) Utilities from International Software Products (ISP). DEMI/DB has largely replaced those tedious tasks, and I would like to detail some of those experiences below – but first, I will explain briefly what DEMI/DB is.

DEMI/DB

Many of you of you may be familiar with ISP's DARSTRAN product for replication. DEMI/DB is a *separate* set of utilities that can be used by companies needing a tool to supplement DARSTRAN. In particular, DEMI/DB can be used to extract, modify, and correct issues with production databases in order to ensure correct data is available to DARSTRAN for the target replicas. It can also be used as a set of stand-alone utilities, which is how we use it at LMC.

How we use DEMI/DB at LMC

DEMI/DB's ease of use makes it flexible enough for us to perform ad-hoc extracts for both specific record types and multiple record types associated with a specific database structure. The extracted data is used as input to other programs to manipulate and/or modify specific fields and to generate reports. The extracts are also used for mass field modification by DEMI/DB and then reloaded into the database.

The following are just a few simple examples of how DEMI/DB is used on a regular basis at LMC.

1. **Account Restructure** – At least once a year, one or more divisions of our company restructure a set of account codes. LMC uses the DEMI/DB Extract utility to extract the data for the affected accounts into flat files. In some instances, the data is modified as it is extracted. If the change is complex, we either use the DEMI/DB Update utility to modify the records or we employ a user written program to read the extract files produced by DEMI/DB and modify the data to produce a new file into the DEMI/DB Import utility.
2. Archive Purge – We also use DEMI/DB to archive and purge records from our database. We have our own utility programs that identify specific records for the Archive and Purge processes and that generate files in DEMI/DB format. By using DEMI/DB to process these files rather than application-specific code, we are able to enhance the performance of our archive and purge processes and eliminate the ongoing maintenance of in-house written programs.
3. Building Test Databases – When we require "cleansed" copies of production data for development or ad-hoc testing, DEMI/DB provides these facilities. Just as we extract data for archiving and purging data, we can build full or partial copies of all or selected structures and records of any database. The ability to replace sensitive data with somewhat meaningless values on the way to creating a copy of the database still provides the structure and relevant data content for application problem analysis or testing of new application code before it goes live.
4. Verifying Integrity – Occasionally, even the best managed databases have issues. We use the DEMI/DB extracts to compare against LMC written programs to verify that the LMC program has properly modified data and if necessary we use the update facilities of DEMI/DB to adjust the database. We also use DEMI/DB to extract all records of a certain type to compare against an SQL extract from a replicated Oracle DB.
5. Daily Extracts – On a daily basis, we have multiple uses for DEMI/DB which eliminates the need to write programs that would require regular maintenance and updates. A few examples of the functions we perform follow.
   a. Determine any out-of-balance PO's within the Consolidated Purchasing System (CPS). Out of balance PO's are defined as those Purchase orders who have one or more line items where the dollar total does not add up to their underlying Material Requisitions (MR). With a few simple control statements, we run the DEMI/DB Extract utility to identify these PO's and provide the data for further analysis.
   b. Search for automatic amendments created by over tolerance receipts – The reporting extract in CPS needs to "know" about any over tolerance amendments created by receiving programs. The DEMI/DB Extract utility is used to locate and extract the data and pass the file to other programs for reporting and analysis.
   c. Load email addresses – In order to keep local e-mail addresses updated in CA IDMS, we run an extract against the global white pages, pass the data

(continued on page 5)

DEMI/DB - A User Experience... cont'd from page 4

to a user written program that generates a file in DEMI/DB Import utility format and then run the DEMI/DB Import utility to update database.

d. "Force" modify buyer email records – the Buyer's Workstation (BWS), an add-on to CPS, needs to be kept in "sync" with CPS. An important part is the buyer's email. This job uses the DEMI/DB Export and Import utilities to extract and do a "dummy" modify of the affected buyer records using the DEMI/DB Import utility. The updates cause our replication system to pick up the changes in CPS and send them to BWS.

6. Ad-hoc query request – Often, our management requests counts of certain record types. It is very easy to write a compact query in DEMI/DB to extract and/or count those records. For example, management wanted a list of all active vendors with valid corporate supplier codes in CPS. All we had to do was prepare these simple parameter statements to select only active vendors in CPS:

```
01,VENDOR-1207,UPD,SWEEP,SMSARE19,D=DD1207


//DD1207   DD  *
        MOVE 4 TO RETURN-CODE.
        IF LOG-DEL-SW-1207 NOT = 'Y' AND 'D'
        IF CORP-SUPP-CODE-1207 NOT = LOW-
VALUES AND SPACES
          MOVE 0  TO RETURN-CODE
          DISPLAY CORP-SUPP-CODE-1207 ' '
VENDOR-NAME-1207
                ' ' VENDOR-CODE-1207.
```

7. **Ad-hoc update request** – Occasionally, anomalies are discovered in the CPS CA IDMS database. Rather than having to write a one-shot program, we can extract the data, and either fix the data in place using DEMI/DB to modify the output file or manually update the data before sending the data to the DEMI/DB import utility to update the data. In order to comply with Sarbanes-Oxley, we use the DEMI/DB Extract utility to dump an image of the data before it is modified. We then use the DEMI/DB utilities to modify the data. As a final step, we use the DEMI/DB Extract utility to get an after image of the data. For example, this code was used to fix "bad" expedite pay codes in CPS:

```
01,PO-ITEM-1210,UPD,SWEEP,SMSARE35,D=DD1210

//DD1210   DD *
        MOVE 4 TO RETURN-CODE.
        IF EXPED-PAY-1210 = X'0000'
          MOVE '3 ' TO EXPED-PAY-1210
          MOVE 0 TO RETURN-CODE.
```

Wrap-Up

As you have seen, DEMI/DB is a great time saver and I highly recommend that you take a look at its capabilities. More information can be obtained from ISP at www.ispinfo.com.

HUMANA ADDS "Z/IIP" TO ITS CA-IDMS PROCESSING

Chris Hoelscher
Senior IDMS and DB2 System Administrator
Humana Inc.

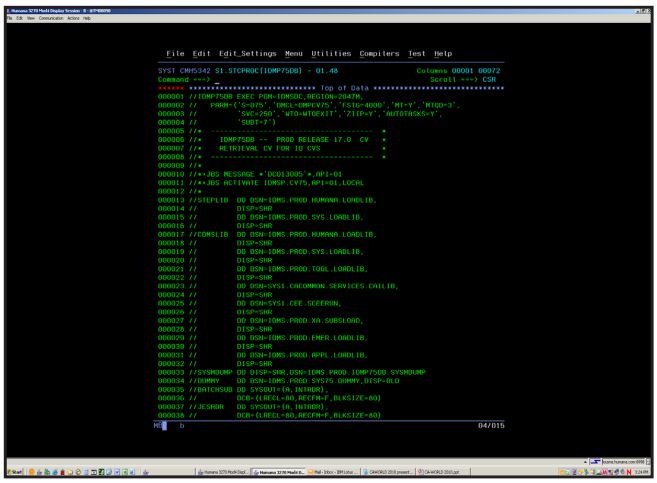
We've all heard about z/IIP processors, but what exactly are they? This presentation will hopefully demystify z/IIP processors so IDMS shops that have access to them can use them and save $ or £ or € or ¥ or Fr or ▯ or €.

What is a z/IIP processor? A z/IIP (short for system/**z** **I**ntegrated **I**nformation **P**rocessor) is available to Z9 & Z10 class processors. A z/IIP processor may be physically identical to a CP (general purpose processor), but is configured through the OS software to appear differently. Currently, each site is limited to one z/IIP processor per CP processor

Why run on a z/IIP processor? IBM will not impose software charges on z/IIP capacity, and requires only a one-time charge for purchase (at substantially less cost than a corresponding CP processor). In other words, execution cycles processed on a z/IIP are ***free*** of (IBM's and others') software mips-based charges.

What runs on a z/IIP processor? Enclave SRB tasks, of course. And *what*, might you ask, is an enclave SRB task? SRB tasks are very lightweight and efficient threads of execution that are available only to supervisor state/privileged mode software, and enclaves are groups of related SRB units "bundled" together. For a more detailed explanation and history of SRB tasks, I highly recommend Peter Morrison's White Paper on the subject at http://www.ca.com/Files/WhitePapers/ziip_exploitation_wp3.pdf . For CA-IDMS, enclave SRB tasks include all system mode work except physical I/O, SVC processing, and user-written exits. IDMS then instructs WorkloadManager to send a certain portion of this eligible workload (currently 30%) to a z/IIP processor.

How can CA-IDMS utilize z/IIP processors? First, migrate to Release 17 or higher (This has not been retrofitted to earlier releases). Next, code ZIIP=Y in your startup PARM= field (this can not be currently specified in #DCPARM).



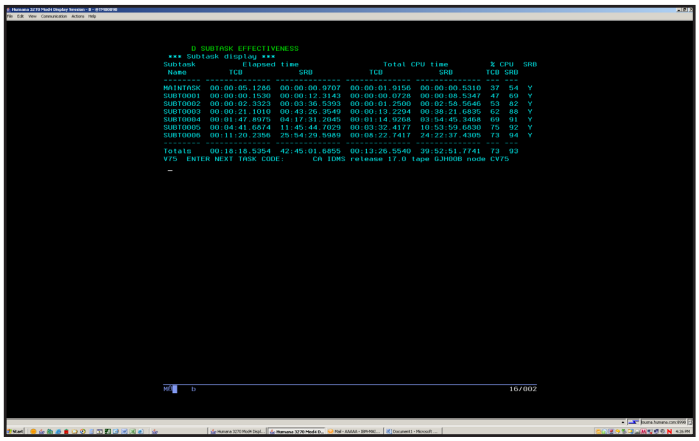
(continued on page 6)

Humana adds "z/IIP"... cont'd from page 5

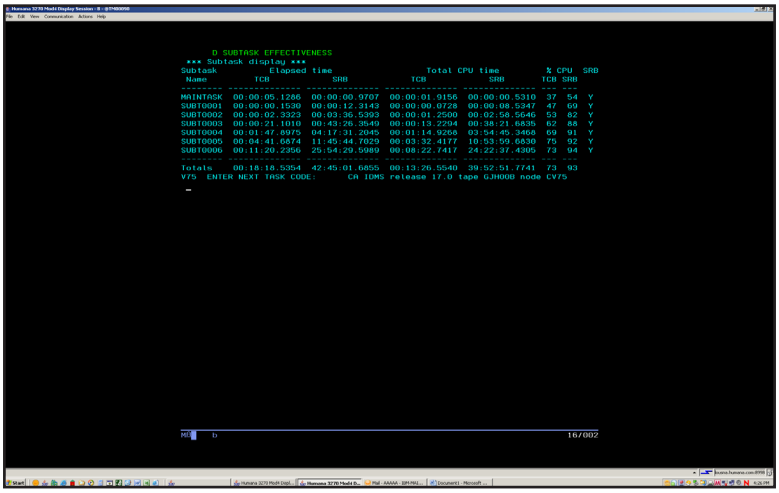
Next, APF-Authorize loadlib(s) from which nucleus modules, line drivers, and service drivers are loaded. Earliest documentation suggested that the loadlib from which RHDCUXIT is loaded needed to be APF-authorized; this has since been shown not to be the case. Keep in mind that only the specific loadlib(s) be authorized; the entire DDNAME concatenation need NOT be authorized. If you specify ZIIP=Y but do not APF-authorize the required loadlib(s), IDMS will inform you of this via message DC016106, and bring up the CV without z/IIP exploitation. On the other hand, if you implicitly or explicitly allow ZIIP=N, but CA-IDMS detects z/IIP processors available, it will inform you that you *are* eligible for z/IIP exploitation via message DC016105.

How can I monitor CA-IDMS utilization of z/IIP processors? First, verify with task DCPROFIL that the CV did in fast start in z/IIP exploitation mode: on the first page of information displayed, to the right of OPERATING SYSTEM: you should see ZIIP=Y. Second, the following DCMT tasks are useful in monitoring z/IIP utilization:

DCMT D SUBTASK EFFECTIVENESS
DCMT D SUBTASK xxx



The column "Total CPU Time/SRB" shows the amount of z/IIP-eligible workload. Compare that value to:



The sum of the "zIIP time" values from all subtasks - this should be 30% of the value from the subtask effectiveness screen – if it is substantially less, then you a) do not have enough z/IIP processors to handle the work IDMS *could* process there, or b) you have the max number of z/IIP processors allowed, but they are busy enough of the time to preclude all z/IIP-scheduled work from executing there. Either way, the value "zIIP on CP time" reflects the missed opportunities. Unfortunately, the sum of these two values determine the 30% threshold; if an SRB enclave is directed to a z/IIP processor, but does not execute there, it counts towards the 30%.

What were Humana's experiences with implementing and monitoring CA-IDMS utilization of z/IIP processors? Humana was very interested in the prospect of utilizing z/IIP processors within CA-IDMS; so much so that in November 2008 we abandoned our nearly-completed migration from release 15 to release 16 and began the migration effort to the newly-available IDMS release 17. We had our first release 17 CV up in December 2008, promoted non-production to release 17 in early March 2009, and production in mid-April. The specific z/IIP implementation steps were quite easy (our system folks APF-authorized the loadlibs, and I made the CV startup changes in under 5 minutes).

As far as monitoring z/IIP utilization, we saw immediate positive results. We support 21 production CVs, two of them serving as read-only Database Owning Region (DORs)(also referred to as back-ends) for both IDMS and CICS Application Owning Regions (AORs)(also referred to as front-ends). One of these read-only CVs in particular processes over 500 million tasks/week, with peak processing of 2500 tasks/second (a whopping 87 CPU hours/week). Of this work, over 99% <!> (86.5 hours) is z/IIP eligible; of the 30% (27 hours) sent to the z/IIP processors, less than 5% (114 minutes) of those enclaves were kicked back to the CP. Our capacity folks inform is that the reduction in software overages costs will save Humana over $120,000 per year.

Where can I find more information about CA-IDMS and z/IIP processors? Please refer to the *CA IDMS r17 Release Summary*, and the *CA IDMS System Operations Guide*. Also, plan to attend CA-WORLD 2010; there are sure to be in-depth presentations on this topic.

THE COBOL XML GENERATOR

by Kay Rozeboom

In the previous issue of "IUA Connections", I explained how to extract data from an XML document, using the COBOL XML parser. In this issue, I will demonstrate how to convert data to XML format, using the COBOL XML generator. We will discuss two examples:

- Example 1 converts a fixed number of fields to XML.
- Example 2 converts a variable number of fields to XML.

Example 1:

Figure 1 shows a simple XML string in its actual format.

Example 1 – XML Output
(data highlighted)

```
<OutputFields><BillingNumber>142A</BillingNumber><Desc
ription>Bob's Bargain Bins and Barns</Description></Ou
tputFields>
```

Figure 1

The data is highlighted in yellow. The non-highlighted characters comprise the XML tags. Quick review: an XML element is composed of three fields: the start tag, the data itself, and the end tag. For example, element "BillingNumber" is composed of:

Start tag = <BillingNumber>
Data = 142A
End tag = </BillingNumber>

The XML in Figure 1 contains "nested elements", also referred to as "sub-elements". Elements "BillingNumber" and "Description" are sub-elements of element "OutputFields".

Example 1 – XML Output
(start-of-element tags highlighted)

```
<OutputFields>
    <BillingNumber>
        142A
    </BillingNumber>
    <Description>
        Bob's Bargain Bins and Barns
    </Description>
</OutputFields>
```

Figure 2

It is possible to convert your data to XML format by coding all of the tags and data fields in working-storage, or by using a series of "string" commands. But IBM has provided a better solution: the XML GENERATE command.

The XML GENERATE command starts with a data structure in working-storage. It converts the field names to start and end tags, and places the field contents between the tags, removing any extraneous white space. Higher data level numbers are converted to sub-elements of lower level numbers. For example, a 10-level field will become a sub-element of a 05-level field.

Figure 2 shows the same XML as Figure 1, but in the more common display format. This time, the start-of-element tags are highlighted. The color-coding of the elements in Figures 3 and 4 will correspond to that in Figure 2.

Figure 3 shows the working-storage data structure that was converted to the XML in Figure 2. Note how the

Example 1 – Working Storage

```
05  OutputFields.
    10  BillingNumber          PIC X(04)   VALUE SPACES.
    10  Description            PIC X(32)   VALUE SPACES.

05  WS-XML-OUT                 PIC X(200)  VALUE SPACES.
05  WS-XML-OUT-LENGTH          PIC S9(8)   COMP VALUE 0.
```

Figure 3

field names have been used for the start and end tags. Note also how the 10-level fields have been converted to sub-elements of the 05-level field.

There are some rules and restrictions to this XML conversion, which are documented in the COBOL manuals. For instance, fields named "FILLER" are ignored.

Figure 4 shows how to code the XML GENERATE command. The first step is to move your data to the working-storage data structure in Figure 3. Then execute the XML GENERATE command. The "ON EXCEPTION" code is executed if an error occurs while

Example 1 – XML Generate Command

```
MOVE DUSYST-BILLING-NUMBER TO BillingNumber.
MOVE DUSYST-DESCRIPTION    TO Description.

XML GENERATE WS-XML-OUT                  <-- Generated XML
  FROM OutputFields
  COUNT IN WS-XML-OUT-LENGTH             <-- Length of generated XML

    ON EXCEPTION
        PERFORM 2100-HANDLE-XML-ERRORS
    NOT ON EXCEPTION
        MOVE WS-XML-OUT TO OUTPUT-RECORD
END-XML.
```

Figure 4

(continued on page 8)

The COBOL XML Generator continued from page 7

generating the XML. The "NOT ON EXCEPTION" code is executed when the command completes successfully. Upon successful completion, WS-XML-OUT will contain the generated XML string, and WS-XML-OUT-LENGTH will contain the length of that string.

Sample Error-Handling Code

```
2100-HANDLE-XML-ERRORS.

    IF XML-CODE = 70          <-- XML return code
        MOVE 0 TO XML-CODE
    ELSE
        COMPUTE WS-XML-DOCUMENT-LENGTH
          = FUNCTION LENGTH (XML-TEXT)   <-- Position of error in XML document
        MOVE WS-XML-DOCUMENT-LENGTH
          TO DISPLAY-XML-DOC-LENGTH
        MOVE XML-CODE TO DISPLAY-XML-CODE
        DISPLAY '--- XML EXCEPTION: ',
                DISPLAY-XML-CODE, ', AT OFFSET: ',
                DISPLAY-XML-DOC-LENGTH
    END-IF.
```

Figure 5

A common mistake is defining WS-XML-OUT large enough to hold only the converted data. You must make it large enough to hold the generated element tags as well.

Figure 5 is a generic XML error-handling routine. This routine is called when an "EXCEPTION" event is encountered by the generate command in figure 4. The "IF" part of the code demonstrates how to bypass selected errors. The "ELSE" part shows how to display the location of the error in the generated XML string.

Example 2:

Example 2 – XML Output
(data highlighted)

```
<OutputFields><CaseNbr>5</CaseNbr><MorePaymentsFlag>NO
</MorePaymentsFlag><PaymentInfo><CreditDate>20070701</
CreditDate><PaymentDate>20071001</PaymentDate><Receive
dAmount>78.10</ReceivedAmount></PaymentInfo><PaymentIn
fo><CreditDate>20070801</CreditDate><PaymentDate>20071
001</PaymentDate><ReceivedAmount>270.00</ReceivedAmoun
t></PaymentInfo></OutputFields>
```

Figure 6

Figure 6 shows another XML string in its actual format. As in Figure 1, the data is highlighted in yellow, and the non-highlighted characters comprise the XML tags. (Notice how the tags take up more space than the data itself. This is not unusual in XML.)

Figure 7 shows the same XML as Figure 6, but in the more common display format. This time, the start-of-element tags are highlighted. In contrast to Example 1, Example 2 has a variable number of fields. Element "PaymentInfo" and its sub-elements can occur 0 to 100 times. (It occurs 2 times in the example.) The color-coding of the elements in Figure 7 corresponds to that in the remaining figures.

Example 2 – XML Output
(start-of-element tags highlighted)

```
<OutputFields>
    <CaseNbr>5</CaseNbr>
    <MorePaymentsFlag>NO</MorePaymentsFlag>
    <PaymentInfo>
        <CreditDate>20070701</CreditDate>
        <PaymentDate>20071001</PaymentDate>
        <ReceivedAmount>78.10</ReceivedAmount>
    </PaymentInfo>
    <PaymentInfo>
        <CreditDate>20070801</CreditDate>
        <PaymentDate>20071001</PaymentDate>
        <ReceivedAmount>270.00</ReceivedAmount>
  </PaymentInfo>
</OutputFields>
```

Figure 7

Figure 8 shows the working-storage data structure that was converted to the XML in Figure 7. Field "PaymentInfo" is defined as an "occurs depending on" field. The COBOL XML generator will use the value of the "depending on" counter (WS-PAYMENT-SUB) to determine how many "PaymentInfo" elements to generate.

Example 2 – Working Storage

```
01  OutputFields.
    05  CaseNbr                PIC X(07) VALUE SPACES.
    05  MorePaymentsFlag       PIC X(03) VALUE SPACES.
    05  PaymentInfo
        OCCURS 0 TO 100 TIMES
          DEPENDING ON WS-PAYMENT-SUB.
        10  CreditDate         PIC X(08) VALUE SPACES.
        10  PaymentDate        PIC X(08) VALUE SPACES.
        10  ReceivedAmount     PIC X(10) VALUE SPACES.

01  WS-XML-OUT                 PIC X(20000) VALUE SPACES.
01  WS-XML-OUT-LENGTH          PIC S9(08) COMP VALUE 0.
01  WS-PAYMENT-SUB             PIC S9(08) COMP VALUE 0.
01  WS-FORMAT-PAYMENT          PIC ZZZZZZ9.99.
```

Figure 8

Figure 9 shows the code that fills in the "PaymentInfo" fields and increments the "depending on" counter. This code is executed 0 to 100 times, depending on how many payment records are found for the selected case number.

Example 2 - Loop

```
------------------------------------------------------
This code is executed once for every payment record found.
------------------------------------------------------

ADD +1 TO WS-PAYMENT-SUB.   <-- "Occurs depending on" counter

MOVE RECEIPT-DATE TO CreditDate (WS-PAYMENT-SUB).
MOVE RECEIVED-DATE TO PaymentDate (WS-PAYMENT-SUB).
MOVE CASE-RECPT-AMT TO WS-FORMAT-PAYMENT.
MOVE WS-FORMAT-PAYMENT
  TO ReceivedAmount (WS-PAYMENT-SUB).
```

Figure 9

(continued on page 9)

The COBOL XML Generator continued from page 7

Figure 10 shows the code that fills in the fields that occur only once, then executes the "XML Generate" command. This is basically the same logic shown in Figure 4 for Example 1.

Example 2 – XML Generate Command

```
-------------------------------------------------------
MOVE CASE-NBR-IN TO CaseNbr.

IF WS-PAYMENT-SUB GREATER THAN 100
    MOVE 'YES' TO MorePaymentsFlag.

XML GENERATE WS-XML-OUT          <-- Generated XML
  FROM OutputFields
  COUNT IN WS-XML-OUT-LENGTH     <-- Length of generated XML

    ON EXCEPTION
        PERFORM 2100-HANDLE-XML-ERRORS
    NOT ON EXCEPTION
        MOVE WS-XML-OUT TO OUTPUT-RECORD
END-XML.
```

Figure 10

If "PaymentInfo" had been defined as simply "occurs 100 times", then 100 "PaymentInfo" elements would have been generated. The first two would have contained data, and the remaining 98 would have been empty elements, consisting of only the element and sub-element tags. That would have added over 10,000 unneeded bytes to the XML string.

Additional resources:

- Both the "COBOL Reference" and the "COBOL Programming Guide" contain detailed information about COBOL XML processing.
- XML error codes are listed, with explanations, in Appendix D of the COBOL Programming Guide.

Biographical note:

Kay Rozeboom is a DBA/Systems Programmer with the State of Iowa. She has 20 years of IDMS experience. Her special interest is in integrating mainframe data and applications with other platforms.

FROM OZIUA - THE AUSTRALIAN IDMS USER GROUP

September Meeting

Last Meeting Wednesday, September 5th, 2009 We had a viewing of a pre-recorded web cast which enlightened us about IDMS Release 17.0 and some of its features that we can look forward to, as well as how Release 17.0 fits in to the CA Mainframe 2.0 environment. Many thanks to CA for its support of the User Group program, and to the attendees who helped to make the session well worth attending.

We tried a new format of holding a single session, at the end of the day, followed by drinks. We didn't get our usual number of attendees so I will need to see whether it was just the way that I advertised the session (it was a bit misleading not emphasizing the Release 17 features enough), or the new format.

A positive that came out of the meeting: you can ask CA to cut you a video of a webcast instead of playing a webcast over the web (I find that at our site there is often a lag between voice and image that can be upsetting). If you cut this to a USB drive or a DVD you can play the Web Cast directly on the machine that's connected to a projector - removing time lag introduced by vagaries of LAN bandwidth and usage when you are trying to view the video. I will try to organize another event before the end of the year.

OZIUA Home Page Relocated

I have made a unilateral, executive decision to change http://oziua.shorturl.com to point to the CA User Groups, Australian IDMS User Group log in page. I have a number of reasons for doing this:

- Too hard to maintain content in two places - one is hard enough
- Once we get enough people "signed up" it will mean that our mailings can be better managed and more highly directed to people who are actually interested
- Help to get our "numbers up" to help to demonstrate to CA that their customers are getting value from the money being invested in the User Group program
- etc, etc, etc !

I hope to move some of the content to the CA User Group site, and the rest of it to the IUA web site at http://www.iuassn.org but we'll see what I am able to achieve, with the help of others, and in the fullness of time.

That's all there is because there is no more – cheers - Gary

Gary Cherlet
Justice Technology Services
President Australian IDMS User Group (OZIUA)

AUTOMATIC RESET & ALERT FOR PROGRAM OUT-OF-SERVICE

by David Matthews

Overview

Programs going out of service should be a very rare event in Production, we think. But for various reasons, it is instead, not uncommon. I had resisted doing anything proactive, because it seemed so unlikely, but a weekend incident persuaded me otherwise.

In our system, the program put out-of-service is often a common module used by multiple other programs. The root cause of the abends taking the program out-of-service is usually a user-error (a bad profile, bad-input, or a fat-finger). The combination of these circumstances means that we want to put the program back in service ASAP, and alert the DBAs to investigate the problem.

This automation does both things.

Technical Approach

The approach taken is simple enough (for a DBA . . . 8^) . . .):

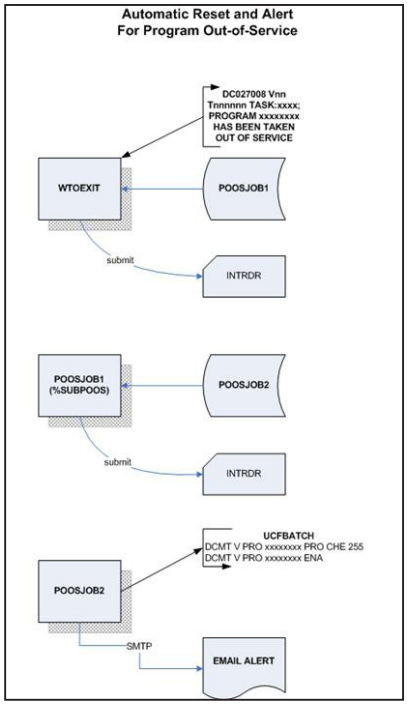
- Update the WTOEXIT to recognize DC027008 -see figure 1
- Insert the message into the JCL Submitted by the WTOEXIT - figure 2
- Parse the message for the program-out-of-service name, insert that name into another JCL and submit it - figure 3
- Reset the program and Alert the DBAs - figure 4
  - Use UCFBATCH DCMT commands to re-enable the program and to raise the Program Check Threshhold (temporarily) to keep it from going right back out of service
  - Send an email (and phonemail) to the DBAs as an alert

Summary

I never wanted to do this before, thinking it would be both a Rube Goldberg mechanism and overkill for a rare problem, but actual events overtook me. What would be nice, is if CA were to include a subset of their Automation product as part of IDMS, as they do for CULPRIT, to help with situations like this.

Editor's Note: this article was submitted by David Matthews from DHL

e-mail:[David.Matthews@dhl.com]



WTOEXIT Logic - figure 1

```
oosMSG   DS    0H                                                    POOS
         OPEN  (JESRDR,(OUTPUT))   JES INTRDR                        POOS
         OPEN  (POOSJOB,(INPUT))   JCL to submit                     POOS
NEXTpoos DS    0H                                                    POOS
         GET   POOSJOB,JCLREC      Read a record                     POOS
         clc   =c'<INSERT>',jclrec ready for insert?                 POOS
         bne   by_insert           no                                POOS
         mvc   jclrec(71),wtomtext yes, insert into jcl              POOS
by_insert ds 0h                                                      POOS
         PUT   JESRDR,JCLREC       Punch it out                      POOS
         B     NEXTpoos              and go for the next one.        POOS
CLOSEOOS CLOSE (JESRDR)                                             POOS
         CLOSE (POOSJOB)                                            POOS
         B     RETURN                                                POOS
```

POOSJOB1 JCL - figure 2

```
000001 //POOSJOB1  JOB  (),'DBA',CLASS=A,
MSGCLASS=X,MSGLEVEL=(1,1)
000004 //* ------------------------------------------------------------
000005 //* SUBMIT  JOB  TO RESET OUT-OF-SERVICE PROGRAM AND
RAISE ALERT
000006 //* ------------------------------------------------------------
000007 //BTSO     EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
000008 //SYSPROC   DD  DSN=IDMSDBA.CLIB,DISP=SHR
000009 //SYSTSPRT  DD  SYSOUT=*
000010 //SYSTSIN   DD  *
000011 PROFILE MSGID
000012 %SUBPOOS IDMSCV
000013 //POOSMSG   DD  *
000014 <INSERT>
000015 //* DC027008 VNN TNNNNNNN TASK:ADS2; PROGRAM XXXXXXXX
HAS BEEN TAKEN
```

SUBPOOS CLIST - figure 3

```
PROC 1 CV DEBUG
CONTROL NOMSG NOFLUSH
IF &DEBUG=DEBUG THEN CONTROL LIST CONLIST SYMLIST
OPENFILE POOSMSG
GETFILE  POOSMSG
SET SYSDVAL=&STR(&POOSMSG)
READDVAL P1 P2 P3 P4 P5 P6 P7 P8
E '&CV..JCL(POOSJOB2)' CN NON NORECOVER
V
C * 999 'XXXXXXXX' '&P7' ALL
SUB
END N
CLOSFILE POOSMSG
EXIT
```

```
000001 //POOSJOB2  JOB  (),'DBA',CLASS=A, MSGCLASS=X,MSGLEVEL=(1,1)
000003 //* ------------------------------------------------------------
000004 //* DC027008 VCC TNNNNNNN TASK:ADS2; PROGRAM XXXXXXXX HAS BEEN TAKEN OUT
000005 //         SET CV=IDMSCV
000006 //UCF      EXEC PGM=RHDCUCFB,REGION=4096K
000007 //STEPLIB  DD  DISP=SHR,DSN=&CV..DBA.LOADLIB
000008 //         DD  DISP=SHR,DSN=CAI.&CV..LOADLIB
000009 //SYSCTL   DD  DSN=&CV..SYSCTL,DISP=SHR
000012 //SYSLST   DD  SYSOUT=*
000013 //SYSIPT   DD  *
000014 DCMT V PRO XXXXXXXX PROGRAM CHECK THRESHOLD 255
000015 DCMT V PRO XXXXXXXX ENA
000016 BYE
000017 //*
000018 //* SUBMIT EMAIL
000019 //*
000020 //EMAIL    EXEC PGM=IEBGENER,REGION=4096K,COND=(0,LT)
000021 //SYSIN    DD  DUMMY
000022 //SYSUT1   DD  *
HELO AEIQ
MAIL FROM:<DBA@company.com>
rcpt TO:< DBA@company.com>
DATA
TO:< DBA@company.com>
FROM: < DBA@company.com>
SUBJECT: PROGRAM OUT OF SERVICE ON AEIQ V20
000025 PROGRAM XXXXXXXX OUT-OF-SERVICE ON IDMSCV. INVESTIGATE.
000026 //SYSUT2   DD  SYSOUT=(A,SMTP)
000027 //SYSOUT   DD  SYSOUT=*
000028 //SYSPRINT DD  SYSOUT=*
000029 //*
```

CA 'ZAPS' ITS MAINTENANCE DELIVERY FORMAT

by Chris Hoelscher
Senior IDMS and DB2 System Administrator
Humana Inc.

What exactly has changed? Starting with IDMS release 17 Service Pack 1, all published maintenance will be delivered as load module replacements (PTFs) rather than load module zaps (APARs). This was done for several reasons: to satisfy customer requests; to allow IDMS maintenance to be CA-MSM eligible; and to ease fix installation syntax so that fixes no longer specify a service pack. And, to be sure, this method is how many other CA and IBM software fixes are delivered.

How is CA implementing this change? For sites beginning their Release 17 adventure at ServicePack1, the changes will be for the most part transparent (all published fixes after base installation will be PTFs (load module replacements). For sites already running 17.0 pre-SP1, it becomes a bit more tricky. CA has created 'replacement' PTFs (load module replacements) for EVERY pre-SP1 APAR (load module zap), which you may chose to apply to supersede the APARs. Why apply these 'replacement' PTFs? All subsequent 17.0 fixes will reference the PTFs, not the APARs, as PREREQs, so having the PTFs installed will make life easier down the road (highway 17?)

How do I find these 'replacement' PTFs? Follow these steps (for MVS sites). This is how I got hold of them:

- sign on to support.ca.com
- select download center
- select published solutions
- select IDMS/DB - MVS
- select 17.0 as release
- select all components
- select OS as operating system
- enter high fix of RO09355 (this is 1 number less than the lowest replacement PTF)
- select 100 results per page
- select 'select all'
- repeat selection process using operating system of z/OS – this will pick up all PIBs for IDMS release 17
- select RO09984 manually – for whatever reason it is not seen by support.ca.com as part of IDMS/DB – MVS
- select 'view solution cart'
- build your downloadable and download (when viewing solution cart, i typically do NOT select "create package")
- unzip your scart0.zip file and upload individual files to mainframe

Philosophically, What does this mean to us? Before this change, fixes were not dependent (or even aware of) other fixes applied to the same module UNLESS they hit the exact same piece of code (to be sure, you would get the other fixes eventually, but under the old system you only needed to be immediately aware of (and test?) the change in code behavior in which you were interested). With this change, however, when installing a fix to solve a specific problem, you will be installing automatically (and need to test for?) any previous (uninstalled) fixes that affect the same module. Is this a bad thing? Probably not, but it IS something to be aware of.

Any caution? – YES. Since published PTFs do not (and should not) reflect non-published changes (such as test fixes or old-style optional APARs that still (and will continue to) contain ZAPs) and will as such overlay them, it is important to first run an APPLY CHECK to see what other already-installed fixes would be overlaid by installing this fix - if those existing fixes include test or optional fixes, they should be RESTORED first, and re-RECEIVED/APPLIED after the published PTF is installed, UNLESS the PTF states that it supersedes the test or optional APAR, in which case no action is needed.

WWW.IUASSN.ORG

YOUR PORTAL TO IUA SERVICES AND IDMS CONTACTS

CA-WORLD 2010 - LAS VEGAS - MAY 16-20

In case you haven't heard, the date and location for CA World 2010 has changed again. The conference has been moved back to Las Vegas, now at the Mandalay Bay Resort & Casino.

There are new dates too, May 16-20, 2010.

The IUA/EIUA and CA anticipate another exciting conference and we are looking for ideas and speakers.

If you have any ideas who you would like to see, or have an idea on topics you would like to see, or if you think you can give a session, please submit a call for speakers form.

Sessions given by IDMS customers are usually the best received and highest rated. We are looking for customers willing to share their expertise and success stories.

CA staff will be willing to assist you with your session.

To submit a session or propose a topic submit a Call for Speaker form at https://www.ca.com/us/Register/form.aspx?CID=139640 (Select *MF Databases* from the Capability Solutions Area list).

If your submission to be a speaker is selected, as a thank you, your CA World 2010 conference registration will be waived. Please submit your proposed sessions by October 30, 2009.

Watch caworld.com for updated information on the conference in the weeks and months to come.

We look forward to seeing you at CA World.

Steve Rundle IUA/EIUA CA-World Liaison

JTS HELP DESK

We had this e-mail come in to the Help Desk the other day from a programmer who went to the manual and solved their problem all on their own - but it seemed to be one of those "handy hints" that was worth passing along. That is about (a) going to the manual, and (b) the particular paragraph which I'm sure many people have either missed, or missed the implications.

From: Application Developer
Sent: Wednesday, 5 August 2009 9:34 AM
To: XXXXXXXXXXXXXXX
Subject: FW: What caused the problem

Hi Help Desk,

Would you like to know the answer to the problem that caused me so much pain and frustration this week and other people's IDMS screens to hang?

I'll tell you anyway because I didn't know this and perhaps you don't either.

The offending statement in a DC COBOL program was

```
OBTAIN MAPE-PERIOD WITHIN MAPE-REASON-X
                   USING WK-MAPE-REASON-X
```

The USING set sort key field in WORKING-STORAGE was:

```
01  WK-MAPE-REASON-X.
   03  WK-MONITOR-REASON-CODE          PIC X(8).
   03  WK-ASSIGNEE-OFFR-PIN            PIC 9(9) COMP.
   03  WK-SUBJECT-ID                   PIC 9(9) COMP.
```

I found the following note in the DML Reference - COBOL manual:

Note: Due to the architecture of the client interface for Advantage CA-IDMS, 256 bytes will be moved regardless of the actual length of the working storage sort key. This additional storage should be accounted for in order to avoid potential program exceptions that can occur. While these exceptions are rare, they are more probable if the sort-key is defined in a FILE or LINKAGE SECTION definition. To avoid this problem, it is recommended that the sort-key be defined in the program's WORKING STORAGE SECTION, padded to a full 256 bytes; and moved in and out of the FILE or LINKAGE SECTION fields.

I changed the working-storage field to be:

```
01  WK-MAPE-REASON-X.
   03  WK-MONITOR-REASON-CODE          PIC X(8).
   03  WK-ASSIGNEE-OFFR-PIN            PIC 9(9) COMP.
   03  WK-SUBJECT-ID                   PIC 9(9) COMP.
   03  FILLER                          PIC X(240).
```

And the problem went away.

I'm sure that there are plenty of COBOL programs out there that may have this problem and they have been lucky enough to not cause loops, or sporadic, seemingly unexplainable errors.

On the same topic

See the following IDMS-L post from Chris Hoelscher - shows the value of watching IDMS-L doesn't it (note subtle plug for IUA from the editor!).

From: IDMS Public Discussion Forum [IDMS-L@LISTSERV.IUASSN.COM] on behalf of Chris Hoelscher [choelscher@HUMANA.COM]
Sent: Wednesday, 11 March 2009 5:06
To: IDMS-L@LISTSERV.IUASSN.COM
Subject: a small DML change for release 17

8.7 FIND/OBTAIN WITHIN SET USING SORT KEY DML Statement A COBOL program containing the FIND/OBTAIN WITHIN SET USING SORT KEY DML statement might compile with a syntax error although it compiled successfully on a prior release. CA IDMS now ensures compliance with the following rules when processing a FIND/OBTAIN WITHIN SET USING SORT KEY DML statement:

(continued on page 13)

JTS Help Desk cont'd from page 12

(*) You cannot specify multiple field names as the sort key in the USING clause.

(*) You must terminate the DML statement with a period or semicolon after specifying the sort key in the USING clause, unless the statement is followed by an ON clause.

The IDMSDMLC precompiler is enhanced to detect extra parameters and issue a syntax error at precompile time.

Note: For more information about the FIND/OBTAIN WITHIN SET USING SORT KEY DML statement, see the CA IDMS DML Reference Guide for COBOL.

In other words ...

```
IF .....
.....
OBTAIN record-name WITHIN set-name
                USING sort element IF DB-REC-NOT-FOUND
  DISPLAY 'whatever'
ELSE
.....
```

will not pre-compile

```
IF .....
.....
OBTAIN record-name WITHIN set-name
                USING sort element; IF DB-REC-NOT-FOUND
  DISPLAY 'whatever'
ELSE
.....
```

will pre-compile

Chris Hoelscher
Senior IDMS & DB2 Database Administrator Humana Inc
502-476-2538
choelscher@humana.com

Editor's Note: For those who watch IDMS-L you would have seen a recent exchange between Chris and Carla Pereira from CA - basically the good news is that there is a fix available - here is a part of the exchange:

From Chris: IS ro12490/1 to be the published fix corresponding to your test fix?

From Carla: Hi Chris - yes RO12491 will be the PTF number which corresponds to TB99850 (which is for z/OS). RO12490 is the z/VSE version of the fix. These PTFs are going through our internal publication process which involves extensive testing by our QA group before being released publicly. This process usually takes about two weeks provided there are no issues found with the PTFs. I expect these PTFs will be available later next week. The PIB you mentioned (RI12747) documents the new error message introduced by the PTFs. Since it doesn't include a fix that needs tested it was released immediately.

That's all folks - cheers - Gary

Gary Cherlet
Justice Technology Services
Department of Justice, SA Government

UPCOMING EVENTS IN EUROPE

- **EIUA IDMS board meeting in Slough (UK) on November 8th**
- **EIUA CA management meeting at the Manor House,Ditton Park (UK) on November 9th**
- **UKIUA IDMS user group event at the Manor House,Ditton Park(UK) on November 10th**
- **BIUA IDMS user group meeting in Brussels on December 11th**

For agenda and registration please contact your local user group chairman Jan.rabaut@sogeti.be for BIUA and EIUA or steve.rundle@bt.com for UKIUA

FRAGMENTS AND RELOCATED RECORDS

One of the most rigidly enforced rules within an IDMS database is that the dbkey of a record is assigned when a record is stored within the database and cannot change as long as that record resides on the database. A dbkey is composed of the number of the page on which a record is stored and a line index number that points the DBMS to information necessary to locate the record occurrence on the database page. This means that when the size of a record occurrence already existing on the database increases IDMS cannot simply move that record to another page if insufficient space exists on the current page since that movement would require a new dbkey to be assigned.

This article describes variable length fragments and relocated records which are structures used by IDMS to acquire the necessary space for these expanding record occurrences without changing a record's original dbkey. Performance implications of these constructs, ways to minimize their creation, and how IDMS attempts to eliminate their presence are also covered.

Fragments

The type of record whose size is most likely to increase after it is stored on a database is one defined as variable length. Records are considered to be variable length when their element definitions include an OCCURS DEPENDING ON clause or the record occurrences are to be compressed. A variable length record (VLR) occurrence will increase in size when as a result of a MODIFY verb an additional reoccurring group is added to the record or changes to the record's data result in the compression algorithm providing a less effective level of compression.

When a record is defined as being variable length a Record Descriptor Word (RDW) is added to the data portion of the record and an additional pointer position is inserted at the end of the record's prefix. The RDW is used to contain the length of the entire data portion of the record occurrence and the additional pointer position is used to support an internal set known as the fragment chain.

VLR Fragment Chain

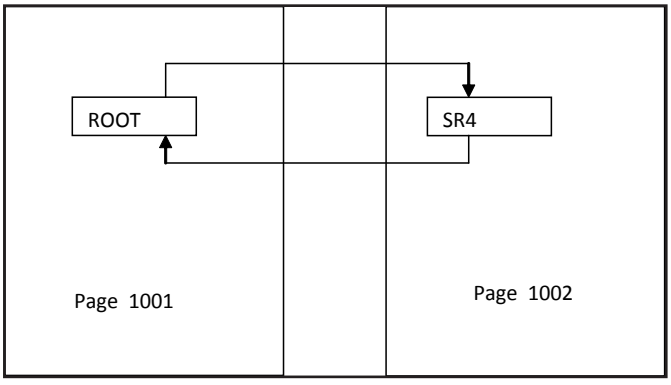


Figure 1

When an existing VLR is to be expanded, and in some cases initially stored, and there is insufficient space to hold the entire record on the occurrence's target page the portion of the record that will not fit will be overflowed to another page within the area. The portion of the record that is on the original page is referred to as the record's 'root' while the overflowed piece of the record is referred to as a fragment and is assigned a record id of 4 (SR4). The root continues to be identified using the record's true record id and acts as the owner of that record's fragment chain with the SR4's being the fragment chain's members.

When IDMS must access the fragmented VLR it will first read the page containing the root of the record as this is the portion of the record referenced by the record's dbkey. IDMS then recognizes that the entire record is not completely contained within the root portion of the record and will commence walking the fragment chain to reconstruct the record within storage. This reconstruction is transparent to the application but the need to read another page to get the record's fragment has a negative impact on the performance of the DBMS. It is possible that the fragment chain may link to multiple fragments, each on a different page. Each page containing a fragment for a VLR record occurrence can be considered to add 1 physical I/O to the amount of work performed by the DBMS to reconstruct the VLR.

Whenever a VLR is defined to a database it is very unlikely that fragmentation of some record occurrences can be completely avoided if the associated applications have the ability to increase the size of these records. However IDMS does provide some mechanisms that will allow a DBA to minimize the number of fragments that may appear within a database. The first thing to do is to insure the proper specification of the MINIMUM ROOT and MINIMUM FRAGMENT clauses on the VLR's RECORD statement within the schema.

MINIMUM ROOT IS RECORD LENGTH should be coded on the schema RECORD statement for all variable length records. This parameter tells the DBMS to insure that the page selected as the target page for a record occurrence has enough space to contain the entire record. Specifying any value other than RECORD LENGTH can result in SR4 fragments being created during the store of the VLR occurrence. In the same manner MINIMUM FRAGMENT IS RECORD LENGTH should always be coded. By coding this you are telling IDMS that whenever it must fragment a VLR during a MODIFY command it must find enough space on an overflow page for the entire fragment. Specification of any other value can result in multiple fragments being created if the database area is tight on space.

Finally, any area that contains a VLR should be defined to have a PAGE RESERVE on its AREA statement within the SEGMENT definition. By specifying a PAGE RESERVE the DBA is directing IDMS to stop adding new records to any page when the amount of free space remaining on that page reaches the specified value. That remaining space can then be used for the expansion of any VLR occurrence on that page which will prevent fragments from being created on a MODIFY command until that reserved space is exhausted.

(continued on page 15)

Fragments and Relocated Records... cont'd from page 14

Acknowledging the fact that VLR fragmentation can have a negative impact on a database's performance, IDMS does make an attempt to condense fragment chains when processing an area in an UPDATE mode. When a fragmented VLR is accessed and the area is in an UPDATE mode IDMS will check the free space on the root's page to see if any space has been made available. If space does exist IDMS will condense as much of the fragment chain as possible back to the original target page and will also condense any other fragments together if space allows on the pages on which fragments exist. Of course the best way to eliminate all fragments within an area is to perform an UNLOAD/RELOAD, DB-REORG, or REORG against that area with a subschema generated from a schema that had MINIMUM ROOT IS RECORD LENGTH specified for the record type.

Relocated Records

Relocated records are created by IDMS during the in-place expansion of a fixed length record or the control length of the root portion of a variable length record. This structure is necessary when such an expansion causes the record (or its control length) to no longer fit on its current page. When this situation is encountered the record occurrence is removed from its current page and replaced with an SR2 record. A new target database page is found for the data record using standard overflow algorithms but the data record is stored on the new page as a type 3 record (SR3) instead of using its original record type designation. The SR2 occurrence contains the relocated record's original record type and the dbkey of the new SR3 record.

When IDMS is directed to access a record at a specified dbkey one of the first things it does is verify the record id of the record on the database page to the type of record requested. If the record ids do not match, IDMS will check the record type on the database page to see if it is a 2 before returning an error status to the application program. If the record is an SR2, IDMS will check the record type indicator within the SR2 to see if it represents the type of record being requested. When the record id in the SR2 matches the requested type the dbkey of the associated SR3 in the SR2 is used to read the SR3 record occurrence. The data record in the SR3 is then returned to the application using its original record type designation.

The only time that SR2/SR3 records may be created for non-SQL defined databases is as a result of the execution of the RESTRUCTURE SEGMENT utility when the record prefix or data length of a record is being increased.

Expansion of records (rows) in an SQL-defined database occurs at various times but is always the result of the execution of a DDL statement. If the prefix portion of a record needs to be expanded due to the addition of a referential constraint an inflight restructure is performed against the database during the execution of the SQL command which changes the SQL schema. If the expansion of a record will not fit during the inflight restructure the necessary relocations occur and the required SR2/SR3 pairs created. If the definition of a table is changed by the addition of new columns no immediate restructure of the data is performed during the modification of the table's definition. However, as these database records are accessed in an update mode the DBMS will append the needed space to any occurrences that existed on the database prior to the change in the table's definition. If this expansion will not fit on the record occurrence's current page relocation occurs and the necessary SR2/SR3 records are created. This means that SR2/SR3 pairs may be created during normal application processing.

In the following example a record type 1024 was expanded and could not fit in the space available on its original page (1001). The record has been relocated to available space on page 1002 and an SR2/SR3 structure created to support the relocated record occurrence.

Relocated Record

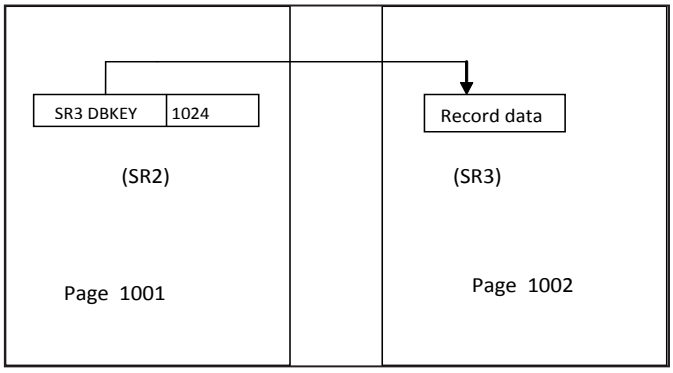


Figure 2

Regardless of the type of IDMS database being used SR2/SR3 records will be removed from the database when a relocated record is accessed and its area is readied in an update mode and there is sufficient space on its original database page to hold the expanded record. In this case the SR2 will be removed from the original page and the record represented by the SR3 will be moved back to that page and re-assigned its original record id. The SR3 is then deleted from its overflow page.

Summary

Variable length fragments or relocated records provide transparency to application programs when the DBMS engine must locate sufficient space for a record by allowing the target record to retain its original dbkey. However this is accomplished through the cost of additional CPU and I/O operations and excessive numbers of these constructs can have a negative impact on an application's performance. DBAs should constantly monitor their database for an increase in these types of system records using the PRINT SPACE or IDMSDBAN utilities and be prepared to reduce their number of occurrences using the UNLOAD/RELOAD, DB-REORG, or REORG utilities.

Dick Weiland is a Senior Systems Engineer for CA IDMS Level II Support working out of the CA Lisle office. He began working with CA IDMS in 1977 as a DBA and joined Cullinet in 1981 in the Field Support organization. He moved to Level II Support in 1988 and is responsible for the DBMS engine and various utilities.

International Chair
Terry Schwartz
Company: Perot Systems
Address: PO Box 269005
Phone: 972 577-3722
Email: terry.Schwartz@ps.net

Secretary/Treasurer
Email Coordinator
Bob Wiklund
Company: Tiburon Technologies
Address: 17101 W. Gable End Lane,
Surprise, AZ 85387
Phone: 623 594-6022
Email: bob_wiklund@tiburontech.com

International Vice Chair
Contributed Software Librarian
Laura Rochon
Company: Ajilon Professional Services
Address: 22 Jolliet, St-Bruno,
Quebec J3V 4Z1 Canada
Phone: 514-943-8290
Fax: 450 441-6880
Email: l.rochon@videotron.ca

European IUA Representative
Steve Rundle
Company: British Telecom BT Group plc.
Address: PP2B33 Angel Centre,
403 St. John Street, London
EC1V 4PL UK
Phone: +44 (0)20 7777 6920
Fax: +44 (0)20 7777 6921
Email: steve.rundle@bt.com

Board Member
Linda J. Casey, PMP, CSM
Company: Managing Member
Run Right, LLC
Email: lindajcasey@runrightllc.com

Board Member
Craig McGregor
Axciom
craig.mcgregor@acxiom.com

Board Member
Diane Montstream
Allen Systems Group
diane.montstream@asg.com

Board Member
Jan Rabaut
jan.rabaut@dexia.be

Editor
Gary Cherlet
Justice Technology Services
South Australian Department of Justice
cherlet.gary@saugov.sa.gov.au

Desktop Publishing
Rebecca Shaw 404 377-6982
shawrh@bellsouth.net

IUA Connections is a quarterly publication of the CA-IDMS Database and Applications User Association (IUA). It is designed to promote its members' objectives. *IUA Connections* is not responsible for the opinions expressed by its writers and editors.

Information User Association
401 N. Michigan Ave.
Chicago, IL 60611-4267
Phone: 312/321-6827
Fax: 312/245-1081

Internet: iua@iuassn.org
Web: http://iuassn.org