CA-ADS® for CA-IDMS® Application Performance

Supplement

I. Dialog Load Module Components

An FDB may contain:

A. Premap Process Element (PME)
B. Record Description Element (RDE)
    1. One for each record used by the dialog (work or internal), its map, and database access
    2. Internal Records
        a) Literal pool
        b) OTB symbol table
        c) OWA ssctrl
        d) VDB date, time
        e) SQL
        f) IRA compute
        g) VRE error-status of record
C. Autostatus Record (ASR)
D. Map
E. Subschema Area Name Table (SSAN)
    1. Every area in the subschema
F. Literal pool 'literals', numbers, dope vectors, etc.
G. Response Element (RSE)
    1. One for each response process
    2. Contains header, name, PFkey, etc. code
H. Ready Area Table (RAT) for each process in a dialog that accesses the database
I. Executable code or a Command Element (CME) for each command
J. Diagnostic Table (if indicated)
K. Symbol Table (if indicated)

II. Application Runtime Variable Storage

A. OTB
    1. Session parameters
    2. One per application
B. OTBX
    1. ADSA-defined application variables
    2. One per application
C. OWA
    1. Only when running
    2. One per application
D. Dialog Variable Storage (VDB)
    1. One per active dialog
E. Subschema Variable Storage (VB50)
    1. Only when running
F. RBB – one for each work, map, or subschema record
    1. One set per application
G. Currency Save Control Block
    1. One per active DB dialog
    2. Only when not running

III. Notify Locks Code Example

A. Basic Values Returned
    00 - No activity
    01 - Record retrieved
    02 - Data modified
    04 - Pointers (prefix) modified
    08 - Logically deleted
    16 - Physically deleted
B. Cumulative Values returned (If more than one activity occurs):
    03 - Record retrieved and data modified
    05 - Record retrieved and pointers modified
    06 - Pointers and data modified
    07 - Record retrieved, pointers and data modified
    09-15 - Logical delete preceded by other activity
    17-31 - Physical delete preceded by other activity
C. Work record layout

```
01 NOTIFY-LOCK-WK
02 LOCKTEST  PIC S9(8)      USAGE COMP.
```

Slide 13: I thought that by extending the Run Unit there would be no need to FINISH on the Link and re-Bind on RETURN to the "linked from" (original) dialog – so I don't understand the "Con"?

This is tricky, and somewhat a question of trade-offs. You are absolutely correct that with an extended run-unit there is no requirement to issue a FINISH & re-BIND on the return to the calling dialog. And it may be simpler to code. However, if the calling dialog manages a lot of different functions within its own code & the code of sub-dialogs it calls, then allowing all of the work to happen in one long transaction (run-unit) may hold locks and resources longer than necessary. So it might be prudent to issue the FINISH/BIND on return from the called dialog, if that does not impact any rollback that might be necessary. So the "Con" here is really the need to consider, when using extended run-units, how long locks and other resources might be held, and then to decide in what circumstances performance would be improved if the calling dialog issued a FINISH/re-BIND after control is returned to it.

Gary adds: and there is a trade-off of individual units of application work be COMMITed independently within the one logical business function! "Partial updates" as application developers call them!

We must keep in mind too that all BIND/FINISH decisions are made for us by ADS in any event … so this is less of a coding issue and more of a performance issue (overhead of multiple FINISH/BINDs in a single transaction – which I have evidence of being very costly indeed) and also the need for more/less granular units of work being COMMITed within the business transaction! Balancing off the "size of the unit of work and the length of time that record locks may be held from beginning to end is a consideration – as mentioned!

My experience is - this is what wins out in the end – the need for an "all or nothing COMMIT"!
Please remember the impact of changing subschema alone can force or prevent Extended Run Units!

Slide 41: Can you please explain the "dirty reads" comment? If another Run Unit has an Exclusive lock (e.g. implicit exclusive from MODIFY verb) then would not DBMS prevent even a "shared retrieval"? My understanding of the DS Retrieval Nolock was that ADS would not reapply Retrieval LOCKs during currency save/restore processing – but that you could not bypass underlying DBMS locking even with combination of Sysgen and ADS compiler specifications?

There's a section in the ADS Reference manual that talks about this. It is indeed possible to completely supress the retrieval locks, so that none are held. And that can result in "dirty reads". This is a term and concept borrowed (I believe) from SQL, where TRANSIENT RETRIEVAL does the same thing. This is what's meant on slide 40 by the line "don't lock". Here's an excerpt from the relevant section in the manual:

"**Suppression of Record Retrieval Locks**

Specifications can be made during dialog compilation to indicate whether or not database record retrieval locks will be held for dialog run units. Retrieval dialogs that **do not update the database** and **do not pass currencies to update dialogs** can be selectively allowed to access database records without locking those records.

Selectively disabling retrieval locks for dialogs allows:

■ Elimination of the overhead of maintaining retrieval locks. This decreases the amount of potential storage and CPU time used by dialogs at runtime.

■ Reduction of the number of db-key deadlocks.

To disable record retrieval locks, you must:

1. **Analyze the dialog in the context of the entire application** to ensure that control and currencies are passed appropriately. A dialog with disabled retrieval locks can pass control and currencies only to a dialog or user program that does retrieval based on these currencies.

2. **Verify the status of the system retrieval locks.** If the mandatory retrieval locks are on, disable the locks at system generation time by specifying RETRIEVAL NOLOCK in the system generation SYSTEM statement.

Note: For more information, see the *CA IDMS System Generation Guide.*

3. **Use the CA ADS dialog compiler or ADSOBCOM** to disable retrieval locking for appropriate dialogs."

Slide 42: Can you please explain these abbreviations – **and how important are they to the presentation** – I'm not sure what needs to be said here? **Area locks (NL, IS, IX, S, U, UIX, X, XR)**

Similarly I'm not sure what is being said in these two lines – or why "record" is used in one place and "row" in another?
Area S (protected retrieval) = S lock on every record
Area X (exclusive) = X lock on every row

These slides are intended to give an overview of locking and the impact it can have on efficiency. Using a least exclusive type lock possible in each dialog will minimize deadlocks & waits, but the programmer must be certain that they're using the appropriate type lock to ensure database integrity for whatever action they intend to perform.

The various lock types are documented in knowledge doc TEC483447 & in the database administration guide. Here are the translations:NL = No lockIS = (retrieval) Intent shareIX = (retrieval) Intent exclusiveS = (retrieval) ShareU = UpdateUIX = Update & intent exclusive UIX+ = No row locksX = ExclusiveThe second Q about the two bullets highlights the fact that a dialog readies an area in a particular status; but contention & waits happen at the record/row level.

The lock that's placed on a particular record/row depends on how the area is readied plus the specific DML commands issued in the dialog. When a dialog readies an area in protected retrieval mode, the DBMS places an 'S' (Share) lock on every record (or row) in the area. If the dialog readies an area in exclusive mode (update or retrieval), then the DBMS puts an X (exclusive) lock on every record or row in the area. If neither protected nor exclusive is specified on the ready statement, then the area is considered to be in shared status, which means different dialogs can place different types of locks on individual records / rows in the area, depending on their needs.

The use of row & record in these two bullets is (I think) intended to point out that the rules of locking are identical for network records & for table rows. They would have been more accurate if both bullets said "record / row" or something similar; that probably wasn't done just for space considerations on the slide.

Slide 45: Sample Code (expanded)

```
!*********************************
!**      PREMAP PROCESS         **
!*********************************
READY.
KEEP LONGTERM ALL RELEASE. <-------- (Clean-up)
OBTAIN CALC CUSTOMER.
KEEP LONGTERM 'CUST' NOTIFY CURRENT CUSTOMER.DISPLAY.
!*********************************
!**   FROM THE RESPONSE PROCESS **
!*********************************
IF NO FIELDS CHANGED THEN <------ (Skip modify logic.)
  DO.
    KEEP LONGTERM 'CUST' RELEASE.
    EXECUTE NEXT FUNCTION.
  END.
 KEEP LONGTERM 'CUST' TEST RETURN NOTIFICATION
  INTO LOCKTEST.
    IF LOCKTEST GT 7 THEN
     DO.
       KEEP LONGTERM 'CUST' RELEASE.
       INIT (CUSTOMER).
       DISPLAY TEXT 'SORRY //RECORD HAS BEEN DELETED'.
     END.
IF LOCKTEST EQ 2 or LOCKTEST EQ 3 or LOCKTEST GT 5  DO.
   OBTAIN CURRENT CUSTOMER.
   KEEP LONGTERM 'CUST' RELEASE.
   KEEP LONGTERM 'CUST' NOTIFY CURRENT
   CUSTOMER.
   DISPLAY TEXT 'RECORD HAS BEEN MODIFIED //
REENTER CHANGES'.
   END.
 MODIFY CUSTOMER.
!************************************************
!** ALLOW USER TO FURTHER MODIFY THE SAME RECORD.**
!************************************************
KEEP LONGTERM 'CUST' RELEASE.
KEEP LONGTERM 'CUST' NOTIFY CURRENT CUSTOMER.
DISPLAY TEXT 'CUSTOMER MODIFIED'.
```

Slide 47: The 3 bullet points describe 3 separate strategies for managing deadlock issues:

1) OBTAIN KEEP EXCLUSIVE
2) LINK TO PROGRAM 'ENQUEUE' – **and presumably afterward to "DEQUEUE" on either side of the process requiring single threading – see presentation referenced at the end?** ENQ/DEQ is on a "Named resource" – depending what you are trying to "single thread" you would assign on a "resource name" – essential to use standards and have well defined code checks to verify these techniques.
3) Invoked dialog to control record update

Can you please expand upon the 3rd strategy – I have seen this used and it did not result in "single threading" – so maybe I'm missing something?

You got the intended concept here- these are different strategies for minimizing deadlocks. The 3rd bullet means that instead of a single large dialog that does many functions, one of which is update, it may be more efficient to pull the update function out into a separate dialog which can ready the area in update mode only for the duration that the smaller dialog runs, and can release the locks as soon as it commits the updates & returns control to the calling dialog. The idea being that the shorter period an exclusive lock is held on a row, the less chance there is for waits &/or deadlocks.

(Caution the implicit FINISH/BIND processing and therefore COMMITs of work done already by the higher level dialog – so careful application design, protocols for "when" to use the lower level dialog for very discrete pieces of work (eg. Before first DML command at higher level or last thing done from higher level)!

Also - the Notify Operator and/or DBA – comment – is this suggesting WRITE LOG MESSAGE with appropriate "routings", for example?

Yes. Or develop some other way for the operator / DBA to monitor the deadlocks & waits that occur. The main point being that unless a site has a production-sized test database, testing may not reveal the deadlocks or waits that occur when a change is moved into production. So someone must be alert to check for these and to work with the development team in making any changes necessary in order to minimize these in the production environment.

Slide 50: Is GI76847 is still the right document to go to? Also – isn't it true that "Activity Logging" only affects "compile time"?

GI76847 is still the correct reference, yes. If folks pull it up on the online support site they will see this line at the top: "12.0 of CA-IDMS and higher. (Currently thru 18.5)". I'm sure it's also good for 19.0 but just hasn't yet been updated to say so.

In terms of your "compile time only" question, it is true that the "Activity logging" parameter will only impact the time it takes to compile a dialog; it has no run=time impact.

The dialog statistics, however, can have a very large run-time impact. If we have to keep statistics for the activity that a dialog accomplishes at run time, the overhead for that will be in addition to the overhead for the actual work that a dialog does. Statistics can be very useful in certain circumstances, but most folks don't look at them all the time for all dialogs,. So we generally recommend that they be turned off at the system level, and sites can use the DCMT VARY STATISTICS command to turn them on or off as needed for debugging or analysis purposes.

Sysgen Recommendations (GI76847)

- ADSO statement parms
  - Activity Log is NO (compile time impact)
  - Dialog Statistics OFF (run time impact)
  - Fast Mode Threshold is OFF
  - Record Compression is OFF
  - Resources are Fixed
  - Storage Mode is Calculated (**maybe** see next)
  - **Tune primary and secondary RBB pools**
  - Maximum Links (10 – value of lower number?)
  - Newpage Mapout No (no literals)

Slide 51: GI76847 is still the right document to go to? Also query regarding "compile time only" impact?
GI76847 is still the correct reference, yes. If folks pull it up on the online support site they will see this line at the top: "12.0 of CA-IDMS and higher. (Currently thru 18.5)". I'm sure it's also good for 19.0 but just hasn't yet been updated to say so. In terms of your "compile time only" question, it is true that the "Activity logging" parameter will only impact the time it takes to compile a dialog; it has no run=time impact.

The dialog statistics, however, can have a very large run-time impact. If we have to keep statistics for the activity that a dialog accomplishes at run time, the overhead for that will be in addition to the overhead for the actual work that a dialog does. Statistics can be very useful in certain circumstances, but most folks don't look at them all the time for all dialogs,. So we generally recommend that they be turned off at the system level, and sites can use the DCMT VARY STATISTICS command to turn them on or off as needed for debugging or analysis purposes.

Regarding Buffer Tuning – see the results from a "before and after" example where RBB's per user dropped from 2 to 1 and RLE chains shortened from 16 to 14 on average. Average RBB usage per user went from 16K to 25K (**cost of only 9K per user**) – there is room in some of the allocated RBB's so there is "room for growth" as application thread extends with LINKs. Consider the impact of shorter RLE chain, reduced need for GETSTG/FREESTG processing – and the imapct in multi-tasking shops where (DC) MP MODE usage can be a bottleneck if you do lots of GET/FREE storage processing – one reason for avoiding "Storage Mode is CALCULATED".

https://communities.ca.com/docs/DOC-15235247

ADSO Buffer Tuning.zip

8K Primary and 8K Secondary Buffer

```
************************************************************
*                  SYSTEM:  20                             *
*               99/327   11:32:30                          *
************************************************************
* Number of LTERMs examined                          509  *
* Number of USERs Signed On                          262  *
************************************************************
* Number of RLEs Allocated to Users                4,221  *
* 31 bit programs - not nucleus or driver            807  *
* 24 bit programs - not nucleus or driver          2,982  *
* Number of Storage Allocations                    4,221  *
*           24 bit Storage (non-XA)                   19  *
*           31 bit Storage (XA)                    4,202  *
* Number of RBBs Allocated                           535  *
*           Primary RBBs                             261  *
*           Primary RBBs less than 50%                 0  *
*           Secondary RBBs                           274  *
*           Secondary RBBs less than 50%             173  *
* Number of VDBs Allocated                           280  *
* Number of $CURCY Allocations                       150  *
* Number of RLTs Allocated                             0  *
* Number of BLLs Allocated                             0  *
************************************************************
```

This output is the last page displayed by the PX52 task.

```
************************************************************
*                  SYSTEM:  20                             *
*               99/327   11:27:31                          *
************************************************************
* Average RLEs per User - all types                   16  *
*         Average RBBs per User                        2  * ←
*         Average RLTs per User                        0  *
*         Average VDBs per User                        1  *
*         Average $CURCYs per User                     1  *
* Average Storage per user - all types           31,472  *
*         Average non-XA Storage per User            35  *
*         Average XA Storage per User            31,436  *
* Average RBB Storage per User                   16,888  * ←
* Average VDB Storage per User                    1,917  *
* Average $CURCY Storage per User                 2,155  *
* Average RLT Storage per User                        0  *
* Average BLL Storage per User                        0  *
* Average "other" Storage per User               10,511  *
************************************************************
```

25K Primary and 12K Secondary Buffer

```
*********************************************************
*                  SYSTEM:  20                          *
*               02/346   16:03:54                       *
*********************************************************
* Number of LTERMs examined                        548  *
* Number of USERs Signed On - using ADS/O          364  *
*********************************************************
* Number of RLEs Allocated to ADS/O Users        5,277  *
* Number of Storage Allocations for ADS/O        5,277  *
*           24 bit Storage (non-XA)                  4  *
*           31 bit Storage (XA)                  5,273  *
* Number of RBBs Allocated                         393  *
*           Primary RBBs                           364  *
*           Primary RBBs less than 50%             114  *
*           Secondary RBBs                          29  *
*           Secondary RBBs less than 50%            27  *
* Number of VDBs Allocated                         394  *
* Number of $CURCY Allocations                     217  *
* Number of RLTs Allocated                           0  *
* Number of BLLs Allocated                           0  *
*********************************************************

*********************************************************
*                  SYSTEM:  20                          *
*               02/346   16:03:54                       *
*********************************************************
* Average RLEs per ADS/O User - all types           14  *
*         Average RBBs per User                      1  *  <-
*         Average RLTs per User                      0  *
*         Average VDBs per User                      1  *
*         Average $CURCYs per User                   1  *
* Average Storage per ADS user - all types      45,336  *
*         Average non-XA Storage per User           50  *
*         Average XA Storage per User           45,285  *
* Average RBB Storage per User                  25,214  *  <-
* Average VDB Storage per User                   2,144  *
* Average $CURCY Storage per User                3,364  *
* Average RLT Storage per User                       0  *
* Average BLL Storage per User                       0  *
* Average "other" Storage per User              14,666  *
*********************************************************
```

Sysgen Recommendations (cont'd)

- System statement parms
  - Retrieval NOLOCK
  - Scratch in XA is YES
  - Reentrant pool sizes

- Do NOT enable ADSALIVE in Production environments
  - **Recent User experience:**
  - Saving of 36% - 48% CPU per task with ADS/Alive disabled
  - Disable by removing USGAFIX and USGADEL as Start-up Autotask
  - ADS/Alive is used for ADS dialog debugging
  - Avoid in PRO

Slide 52: Do NOT enable ADSALIVE in Production environments → Recent User experience: There appears to be on average a saving of 36% - 48% CPU per task with ADS/Alive disabled in PRO. ADS/ALIVE is disabled in PRO by removing USGAFIX and USGADEL as Start-up Autotask.