CA ADS™ Application Performance: Locking Considerations

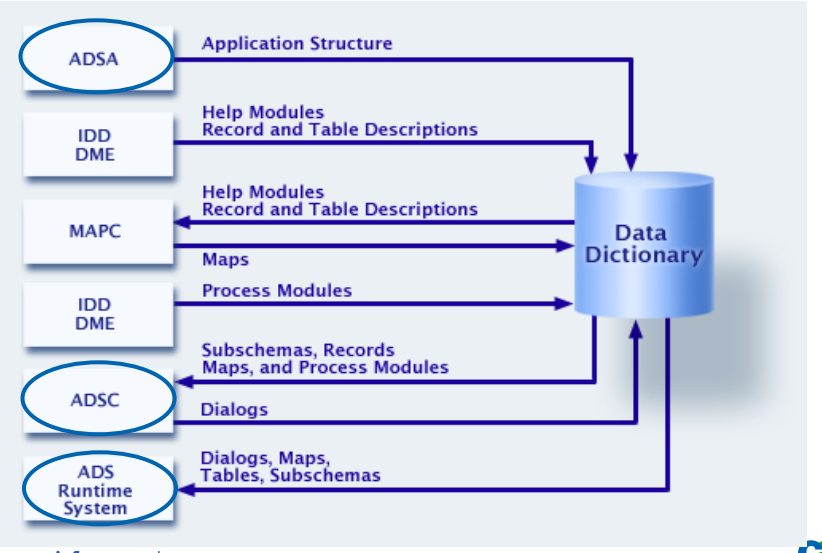
Cal Domingue

maymainframemadness 2012

abstract

— The CA ADS™ family of tools is a powerful and fast way to create runtime applications. Efficiency and performance are important considerations in a production environment. This webcast will discuss locking factors that affect the response time and throughput of runtime performance.

CA ADS family of tools



locking considerations

- Locks control simultaneous access
- Prevent overlapping updates, broken sets
- Deadlock: shared access, exclusive upgrade
- Wait: need lock, a competing lock exists
- Locking goals:
  - Minimize deadlocks, waits
  - Maximize data integrity
- Options:
  - Don't lock
  - Prevent simultaneous access
  - Monitor access, test, manage

area locking mode options

Most Restrictive

Least Restrictive

Usage mode	Qualifier
>Update	>Exclusive >Protected >Shared
>Retrieval	>Exclusive >Protected >Shared
>Transient Retrieval	
>Nolock	

don't lock: RETRIEVAL NOLOCK

- Intended for dialogs to access static data only
- Possible for dialogs with no database update
- Allowed by SYSTEM statement
- Implemented by dialog
  - Dialog Options screen in ADSC
  - Specify RETRIEVAL LOCK: NO
- Saves storage for locking control
- Improves performance
- Use with caution, may result in 'dirty reads'

locking within the run-unit

- Set locks explicitly for the duration of R-U
  - KEEP command or keyword (shared lock)
  - KEEP EXCLUSIVE (exclusive lock)
- Locks set implicitly by ready mode & DML
  - Subschema default for area
  - Default: shared retrieval
- Area locks
- Row locks
  - Shared (upon read) enable retrieval, not update
  - Exclusive (upon update) prevents other access
  - Area S (protected retrieval) = S lock on every record
  - Area X (exclusive) = X lock on every row

locking across run-units

- KEEP LONGTERM (NOTIFY) Lock options
- Can be used to single-thread
- Two common uses during a pseudo-converse
  - Gain exclusive control of a frequently updated record
  - Monitor updates to infrequently updated records

locking across run-units (cont.)

- KEEP LONGTERM <longterm-id> RELEASE
  - Lock already released, yields error 1521
  - Releases all long-term locks linked to lterm
  - If no locks held, still returns 0000 status
- KEEP LONGTERM ALL RELEASE
  - Normal exit from application
  - Code this at application entry point, to clean up after a possible previous abend
- Consider using NOTIFY
  - Create a work field to hold the notification
  - Test against the notification field
  - Code example follows

prevent access by others

Premap Process:

```
OBTAIN CALC CUSTOMER .
KEEP LONGTERM LOCK-ID EXCLUSIVE        CUSTOMER.
DISPLAY.
```

Response Process:

```
MODIFY CUSTOMER.
KEEP LONGTERM LOCK-ID RELEASE.
DISPLAY MSG TEXT 'Customer Modified'.
```


KEEP LONGTERM considerations

- Lock attached to the logical terminal (LTERM)
- **Not** released by COMMIT
- RELEASE required (often forgotten)
  - When the user signs off
  - DCMT V LTERM <ltermid> RES DELETE.
- Managed by lock ID: 1 to 16 characters
  - A constant in single quotation marks
  - A variable field containing the lock ID
- Must be unique for the record type

notify locks code example
background

- Basic Values Returned
  - 00 - No activity
  - 01 - Record retrieved
  - 02 - Data modified
  - 04 - Pointers (prefix) modified
  - 08 - Logically deleted
  - 16 - Physically deleted

notify locks code example
background

- Cumulative Values returned (If > 1 activity)
  - 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
- Work record layout
  - 01 NOTIFY-LOCK-WK
  - 02 LOCKTEST PIC S9(8) USAGE COMP.

notify locks example
code sample – page 1 of 3

```
!*********************************
!**     PREMAP PROCESS        **
!*********************************
READY.
KEEP LONGTERM ALL RELEASE. <-------- (Clean-up)
OBTAIN CALC CUSTOMER.
KEEP LONGTERM 'CUST' NOTIFY CURRENT CUSTOMER.
DISPLAY.
```


notify locks example
code sample – page 2 of 3

```
!*********************************
!**   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.
...
```


notify locks example
code sample – page 3 of 3

```
...
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'.
```


handling deadlocks

- Avoid by single threading
  - OBTAIN KEEP EXCLUSIVE
  - LINK TO PROGRAM 'ENQUEUE'
  - Invoked dialog to control record update
- Avoid by using NOTIFY Locks
  - Test for other update first
- Recover from them with automatic recovery
  - ALLOWING ('0329')
  - FIND CURRENT ALLOWING ('0329') on control commands
  - Notify Operator and/or DBA

session summary: minimize locking problems

- Tune databases (page & area size, indexes)
- Use least exclusive usage mode necessary
- Small transactions (recovery unit of work)
- Efficient dialogs (testing)
- No conversational programming with linked-to programs
- No locks held across pseudo-converse

Questions

http://Support.ca.com

(856)273-3411

maymainframemadness 2012

thank you

maymainframemadness 2012

legal notice

THIS PRESENTATION IS FOR YOUR INFORMATIONAL PURPOSES ONLY. CA assumes no responsibility for the accuracy or completeness of the information. TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. In no event will CA be liable for any loss or damage, direct or indirect, in connection with this presentation, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised of the possibility of such damages.

Certain information in this presentation may outline CA's general product direction. This presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA's sole discretion.

Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA may make such release available (i) for sale to new licensees of such product; and (ii) in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis.