CA IDMS™ Deadlock Analysis

Dick Weiland
CA Technologies

IUA/CA IDMS™ Technical Conference May 7-11, 2018

Abstract

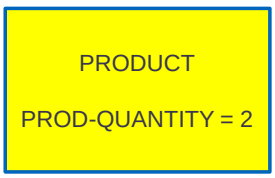
- This session provides an explanation of the locking process and the potential deadlock situations that may happen. The material presents the messages generated when a deadlock occurs and their use in identifying the underlying cause of the deadlock scenario. Finally some general application design recommendations are discussed to minimize the occurrences of deadlocks.

Agenda

1. LOCKING FUNDAMENTALS
2. DEADLOCK SCENARIOS
3. LOCKING ERROR MESSAGES
4. DEADLOCK ANALYSIS
5. GENERAL RECOMMENDATIONS
6. QUESTIONS AND ANSWERS

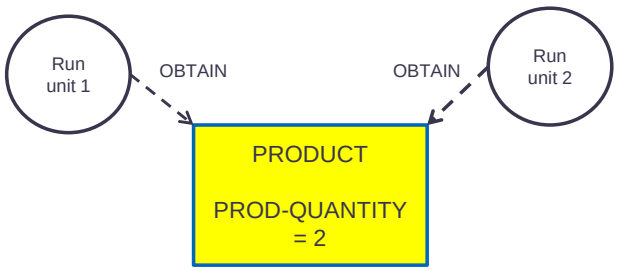
Locking Fundamentals
Why do we need locking?

- Our database contains a record type named PRODUCT which contains a field name PROD-QUANTITY
- PROD-QUANTITY represents the quantity of this product that is in inventory and available to be shipped



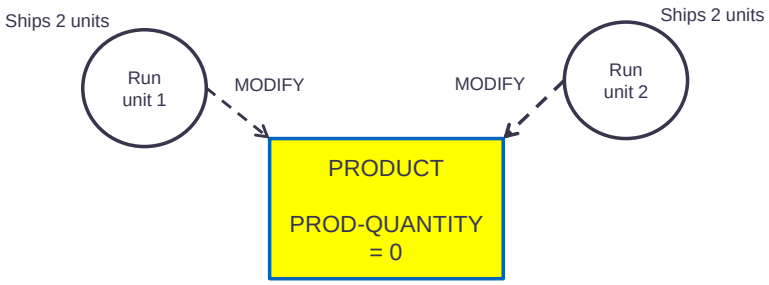
Locking Fundamentals
Why do we need locking?

- Two run-units simultaneously access this occurrence of the PRODUCT record with the intent of allocating 2 units of the product for shipment to a customer.
- Each run-unit sees that 2 units are available and each proceed to update the record occurrence



Locking Fundamentals
Why do we need locking?

- Without locking, each run-unit thinks it has allocated 2 units of the product for shipment to its associated customer but only a total of 2 units exist
- The PROD-QUANTITY field is set to zero although 2 units more than available were allocated for shipment

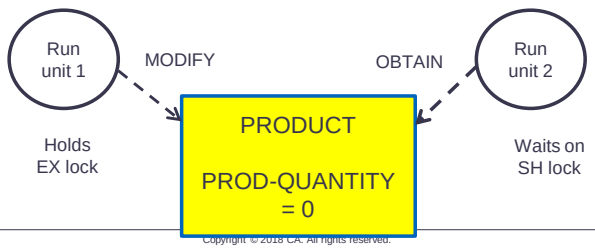


Locking Fundamentals
Types of Record Locks

- CA IDMS maintains two basic types of record locks
  - SHARED locks
    - Also known as RETRIEVAL locks
    - Multiple run-units can read a record occurrence and each can hold a SHARED lock on that occurrence
    - No run-unit may acquire a SHARED lock on a record occurrence on which another run-unit has an EXCLUSIVE lock
  - EXCLUSIVE locks
    - Also known as UPDATE locks
    - Only one run unit can concurrently hold an EXCLUSIVE lock on a record occurrence
    - No run-unit may acquire an EXCLUSIVE lock on a record occurrence on which another run-unit has a SHARED lock

Locking Fundamentals
Locking Contentions

- Locking contention occurs in the following scenario
  - Run-unit 1 reads the PRODUCT record causing a SHARED lock to be set
  - Run-unit 1 then modifies the PRODUCT record causing its SHARED lock to be upgraded to an EXCLUSIVE lock
  - Before run-unit 1 issues a FINISH or COMMIT run-unit 2 attempts to read the PRODUCT record occurrence which tries to establish a SHARED lock on the occurrence. This causes run-unit 2 to go into a wait until it can get the SHARED lock.



Locking Fundamentals
Locking Contentions

- When run-unit 1 issues a FINISH or COMMIT it will release its EXCLUSIVE lock on the record occurrence
- Run-unit 2 can then acquire the SHARED lock and its OBTAIN command can be completed
- Run-unit 2 sees the contents of the record as they were set by run-unit 1's MODIFY command

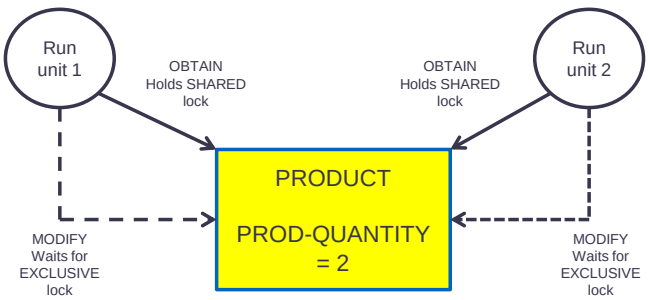
Deadlock Scenarios

- Based on the sequence in which run-units access database records it is possible that two or more run-units each hold locks on record occurrences required by the other run-unit
- When this occurs a deadly embrace or deadlock occurs and one of the run-units must be abended

Deadlock Scenarios (cont.)

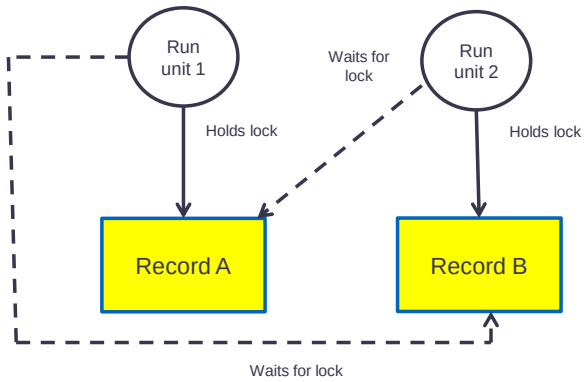
- A deadlock occurs in our example with the following sequence of events
  - Run-unit 1 issues an OBTAIN against a PRODUCT occurrence causing a SHARED lock to be set
  - Run-unit 2 issues an OBTAIN against the same PRODUCT occurrence causing another SHARED lock to be set on the record
  - Run-unit 1 issues a MODIFY against the record occurrence and goes into a wait until run-unit 2 releases its SHARED lock
  - Run-unit 2 issues a MODIFY against the same occurrence and goes into a wait until run-unit 1 releases its SHARED lock
  - A deadlock condition exists between run-units 1 and 2 and must be resolved by abending one of the run-units

Deadlock Scenarios (cont.)



Deadlock Scenarios (cont.)

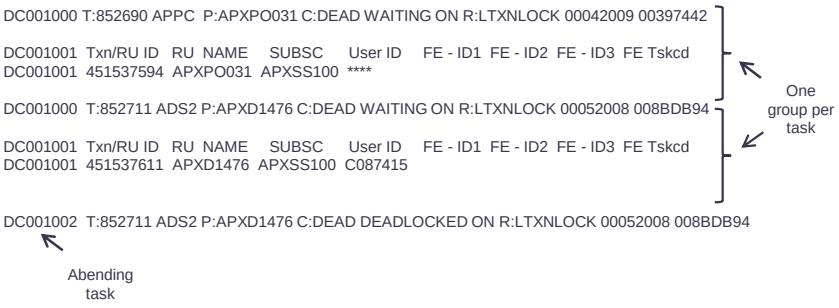
- Deadlocks can occur between multiple run-units and involve multiple record types, record occurrences, or other resources



Locking Error Messages

- When a deadlock condition occurs, a cluster of messages will be written to the CV's log
- One DC001000 for each task involved in the locking contention
- Optionally DC001001 messages can be generated providing additional information about the locking contention
  - Generated when DEADLOCK_DETAILS=ON specified in the CV's SYSIDMS file
  - Two lines per task
    - One line containing headers
    - Second line containing the actual data
- A DC001002 message indicating the task that was aborted as part of the deadlock resolution

Locking Error Messages (cont.)



Locking Error Messages (cont.)

- Each DC001000 message represents a task that is a participant in the deadlock and is waiting for a resource
- The first 4 fields of the message provide:
  - The task id of the waiting task
  - The task code that invoked the task
  - The program name that is being executed
  - The status of the task

Locking Error Messages (cont.)

- The last field following the literal 'R:' provides the resource upon which the task is waiting

R: <Type><wwwwxxyy zzzzzzzz>

Type - LTXNLOCK Local transaction lock
GTXNLOCK Global transaction lock
PAGELOCK PAGELOCK
<wwwwxxyy zzzzzzzz>

<wwww> Page group
<xx> Code identifying the value in **<zzzzzzzz>**

'00' <zzzzzzzz> contains a dbkey (Record lock)
'20' <zzzzzzzz> contains a page number (Space Management lock)
'80' <zzzzzzzz> contains an area's low page (Area lock)
'C0' <zzzzzzzz> contains an area's low page (Transient lock)

'01' <zzzzzzzz> contains a page number (Page lock)

<yy> Dbkey radix

Locking Error Messages (cont.)

- A DC001001 message provides further information to assist in the description of the task described in the preceding DC001000 message
  - Transaction id
  - Program name found in the SUBSCHEMA-CONTROL when the run-unit was bound
  - Subschema to which the run-unit is bound
- One DC001001 message will be displayed for each run-unit associated with the waiting task
- A DC001002 message is generated to identify the task that was aborted to resolve the deadlock
- The contents of a DC001002 is the same as the associated DC001000 but the specified resource can be interpreted as the resource that was the final item completing the deadlock scenario

Deadlock Analysis

- It is extremely rare that the resolution of deadlocks can occur by changing any system-level definitions within a CV
- However if a CV is experiencing excessive deadlocks the CV's SYSGEN should be examined for the following statements
  - DEADLOCK DETECTION INTERVAL IS 1
  - TICKER INTERVAL IS 1
- Specifying a value of '1' for each of these options will ensure that CA IDMS will search for and resolve deadlock conditions as soon as possible

Deadlock Analysis (cont.)

- In any environment that permits concurrent updating of the database it is to be expected that deadlocks will occur
- As transaction volume increases so does the likelihood that points of contention will occur
- Deadlock conditions are almost always the result of the database design and timing issues associated with the way the application navigates the database
- Resolution will usually require database design and/or application changes to reduce the point of contention

Deadlock Analysis (cont.)

- Start your deadlock analysis by scanning the DC001000 and DC001002 messages looking for repeating values
- If a single dbkey is present in a large percentage of the deadlocks you have probably located a major point of contention within the environment
- The DC001000/DC001002 messages do not tell you the type of record represented by the dbkey
- Use the PRINT PAGE utility to print the specified page and to identify the type of record involved
- The DC001000 and DC001001 messages contain the program and run-unit names that were processing the database at the point of the deadlock

Deadlock Analysis (cont.)

- The following innocent looking code can result in serious deadlock conditions which would be reflected by a repeating dbkey in the associated deadlock messages

```
OBTAIN CALC ORDER-CTRL.
PERFORM IDMS-STATUS.
MOVE NEXT-ORD-NUM TO ORDER-NUMBER.
ADD +1 TO NEXT-ORD-NUM.
MODIFY ORDER-CTRL.
PERFORM IDMS-STATUS.
```

Deadlock Analysis (cont.)

- The ORDER-CTRL record is what is known as a One-Of–A-Kind (OOAK) record
- Every time that an order is added to the database it is necessary to read the single ORDER-CTRL record to get the next order number to be assigned and to modify the record
- Large numbers of transactions concurrently adding orders can generate excessive deadlock occurrences
- One possible resolution would be to develop an alternate method to generate an order number
- Another option would be to single-thread this code using ENQUEUE/DEQUEUE commands

Deadlock Analysis (cont.)

- The problem code in our example could be single-threaded as seen below

```
ENQUEUE WAIT CTRL-LOCK LENGTH 8.
 OBTAIN CALC ORDER-CTRL.
PERFORM IDMS-STATUS.
MOVE NEXT-ORD-NUM TO ORDER-NUMBER.
ADD +1 TO NEXT-ORD-NUM.
MODIFY ORDER-CTRL.
PERFORM IDMS-STATUS.
DEQUEUE CTRL-LOCK LENGTH 8.
```

Deadlock Analysis (cont.)

- In some scenarios the repeating resource may not represent a dbkey but a page number
- If the value of the <xx> portion of the DC001000 message is x'20' there is a contention attempting to manipulate the amount of free space on the specified database page
- Use the PAGE PRINT utility to list the specified page(s) to determine the type of records that reside on the page
- Use this information to identify the associated logic in the programs specified on the DC001000 and DC001001 messages

Deadlock Analysis (cont.)

- Common causes of this type of lock contention are the following
  - Large volumes of records written to a single DC Queue using a PUT QUEUE command
  - Large numbers of records defined as DIRECT are added to an area always using -1 as the suggested dbkey
  - Many records always added to the same location within an index
- Single-threading these functions or finding alternative database structures is required to resolve these types of contentions

Deadlock Analysis (cont.)

- If the value of the <xx> component of the DC001000 message is an x'80' you have contention on the usage mode of a database area
- Use the page group and page number to identify the area that is involved
- Inspect the programs specified on the DC001000 messages to determine how they ready the area
- The common problem in these cases is programs that attempt to ready an area in a PROTECTED or EXCLUSIVE mode in an environment where concurrent processing is occurring
- Modify the affected programs to used a SHARED version of the READY verb

Deadlock Analysis (cont.)

- In some scenarios the repeating value within the DC001000 and DC001001 may be a program or run-unit names
- This is typically an indication that there is program logic within these programs that causes contention in the sequence in which various records are accessed
- Identify the records and page numbers associated with the DC001000 and DC001001 messages and use PRINT PAGE to identify the types of records involved

Deadlock Analysis (cont.)

- Identify the most common sequence of DML commands within each specified program and look for those points where the various program logic intersects on the record indicated on the deadlock messages
- These types of scenarios are best addressed by application staff that is familiar with the logic of the programs involved in the deadlock
- Resolution of these types of deadlocks depends on the offending logic discovered and the requirements of the application
- These types of deadlocks tend to be sporadic and usually do not generate large numbers of concurrent deadlock occurrences

Deadlock Analysis (cont.)

- JREPORT 008 (Journal Detail Report) can be a useful resource to assist in determining the sequence of DML commands in these types of deadlocks
- Limit your JREPORT 008 execution to those transaction numbers listed on the associated DC001001 to minimize the generated output
- The report will show the DML commands and their affected database records in the sequence they occur
  - This will show you how the deadlock occurred

General Recommendations

- Each deadlock scenario represents a unique set of conditions but there are a number of considerations that should be observed to minimize their occurrence
- Avoid running batch updates during online windows
  - Exclusive record locks are not released until IDMS is explicitly told to do so through FINISH or COMMIT verbs
  - Batch update run-units should include frequent COMMIT verbs to periodically release locks
- Minimize the amount of work performed by a single online transaction

General Recommendations (cont.)

- Avoid creating database 'hotspots'
  - Locate and eliminate single locations within the database that multiple transactions must update concurrently
    - OOAK records
    - DIRECT records stored using a suggested dbkey of -1
    - A DC queue
    - Index structure where the same or very similar key value is always inserted
- Avoid retaining resources across transactions or interactions with a terminal operator
  - Longterm locks can be held across transactions and can lock a dbkey for long periods of time
    - Their usage should be used only when absolutely necessary
  - Don't use conversational tasks that keep update run-units open while waiting for the terminal operator's response

General Recommendations (cont.)

- Stress test changes prior to production implementation
  - Any modification that changes the sequence or volume of updates to database records should be stress tested at the largest expected volume of concurrent transactions
  - Use a third party product for your stress test such a TPNS from IBM
- Insure that deadlocks are resolved as quickly as possible
  - DEADLOCK DETECTION INTERVAL IS 1
  - TICKER INTERVAL IS 1

Summary

- Deadlock analysis is more of an art than a science which requires the participation of a site's application staff and not just the Database Administrator
- Although the DBA may do the initial analysis of the deadlock messages generated by CA IDMS it will usually require the application developer's specialized knowledge of the application code to interpret, identify, and correct any underlying problems

FOR INFORMATION PURPOSES ONLY

Terms of this Presentation

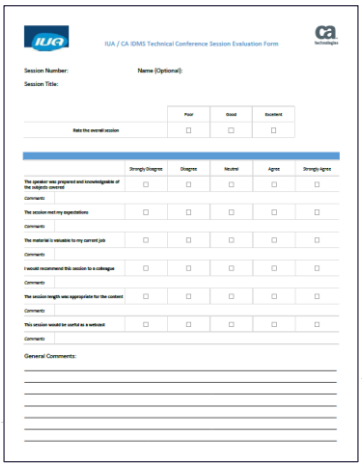
This presentation was based on current information and resource allocations as of May 2018 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations 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 will make such release available (i)
for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and 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. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.

Certain information in this presentation may outline CA's general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this presentation "as is" without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised in advance of the possibility of such damages. CA confidential and proprietary. No unauthorized copying or distribution permitted.

Questions & Answers

Please Complete a Session Evaluation Form

- The number for this session is **D07**
- After completing your session evaluation form, place it in the envelope at the front of the room



IUA / CA IDMS Technical Conference Session Evaluation Form

Session Number: Name (Optional):

Session Title:

	Poor	Good	Excellent
Rate the overall session	☐	☐	☐

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The speaker was prepared and knowledgeable of the subjects covered	☐	☐	☐	☐	☐
Comments:					
The session met my expectations	☐	☐	☐	☐	☐
Comments:					
The material is valuable to my current job	☐	☐	☐	☐	☐
Comments:					
I would recommend this session to a colleague	☐	☐	☐	☐	☐
Comments:					
The session length was appropriate for the content	☐	☐	☐	☐	☐
Comments:					
This session would be useful as a webcast	☐	☐	☐	☐	☐
Comments:					

General Comments: