All About Deadlocks

Arup Nanda

Longtime Oracle DBA

Some Questions

```
ORA-00060 Deadlock Occurred
```

- What's a Deadlock
- How can I prevent it
- Why would an INSERT cause deadlock
- Why would I need to index FK columns
- Is ON DELETE CASCADE FK constraint a good idea?

Objectives

- Why Deadlocks Occur
- Primer on Oracle Latching, Locking
- How to Interpret Deadlock Traces
- Various Cases of Deadlocks
- Some Rare Cases from My Experience

What's a Deadlock (ORA-60)

- When two sessions each lock the resource requested by the other
- Oracle automatically detects the deadly embrace and breaks it by forcing one transaction to end.

Deadlock Scenario

Session 1	Session 2
Update Row1 Does not Commit	
	Update Row2 Does not Commit
Update Row2 Waits on TX Enqueue	
	Update Row1

Types of Enqueues

- TM (metadata related)
  - When DMLs execute, the table structure should remain the same
  - TM protects against DDLs
- TX – when a row is locked

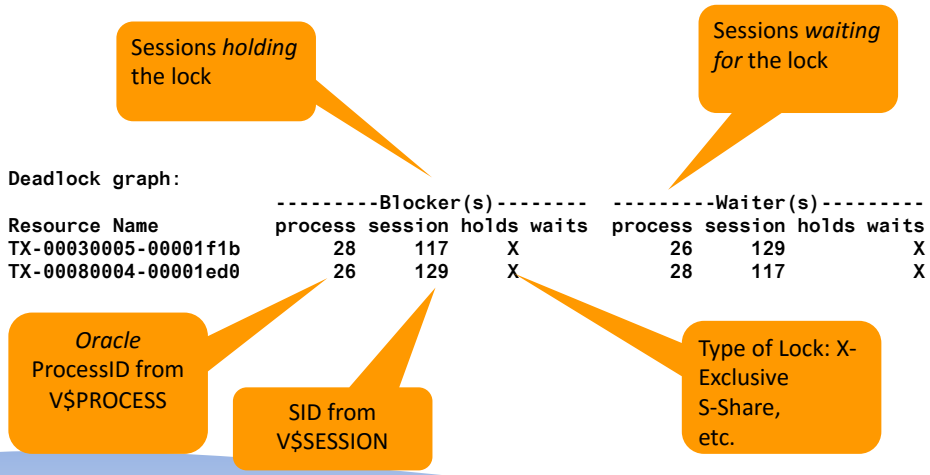
Anatomy of a Deadlock Trace

- Shows up in alert Log:

```
ORA-00060: Deadlock detected. More info in file
  /opt/oracle/diag/rdbms/odba112/ODBA112/trace/ODBA
  112_ora_18301.trc.
```

- See the attached Word Document Showing the Trace File with Annotations.

Deadlock Graph



Row Information

Obj# in HEX

```
Rows waited on:
  Session 117: obj - rowid = 00013923 - AAATkjAAEAAAAKGAAB
  (dictionary objn - 80163, file - 4, block - 646, slot - 1)
  Session 129: obj - rowid = 00013923 – AAATkjAAEAAAAKGAAA
 (dictionary objn - 80163, file - 4, block - 646, slot - 0)
```

```
select object_name, owner
from dba_objects
where data_object_id = 80163

OBJECT_NAME OWNER
----------- ------------
DLT1        ARUP
```

```
select col1
from arup.dlt1
where rowid =
   'AAATkjAAEAAAAKGAAA'

      COL1
----------
         1
```


Process Information

Calling User

```
Session 129:
  sid: 129 ser: 31882 audsid: 202994 user: 90/ARUP flags:
0x45
  pid: 26 O/S info: user: oracle, term: UNKNOWN, ospid:
24275
    image: oracle@oradba1 (TNS V1-V3)
  client details:
    O/S info: user: oracle, term: pts/6, ospid: 24274
    machine: oradba1 program: sqlplus@oradba1 (TNS V1-
V3)SQL for the other session
    application name: SQL*Plus, hash value=3669949024
  current SQL:
  update dlt1 set col2 = 8 where col1 = 1
```

Is That All?

- NO!
- Other types of locks manifest differently, creating deadlocks.
  - Difficult to Identify
- Other Causes
  - ITL Waits
  - Bitmap Index Update
  - Direct Path Load
  - Overlapping PK Values

Locks and ITL

- A detailed discussion of ITLs is found at http://arup.blogspot.com/2011/01/more-on-interested-transaction-lists.html
- Each ITL takes up 24 bytes

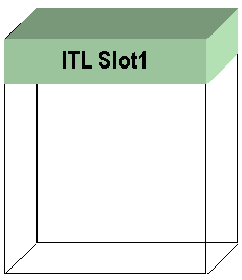


Figure 1

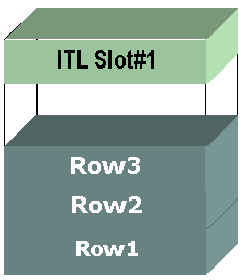


Figure 2

Locking for Sessions

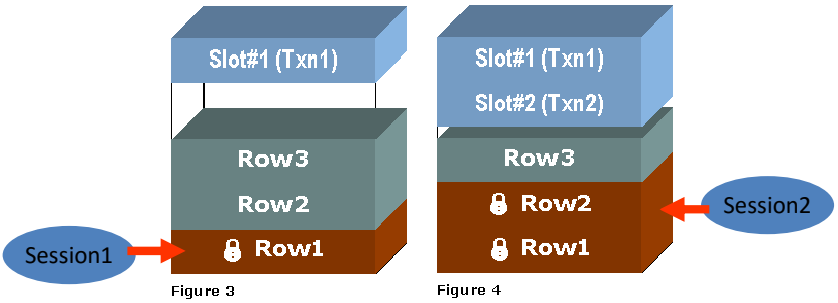


Figure 3 Figure 4

At this point, there is no more room in the block to create another ITL entry; so another lock will be impossible.

Checking for ITL Shortage

- The Query Gets it:

```
select owner, object_name, value
from v$segment_statistics
where statistic_name = 'ITL waits'
and value > 0
```

- Here is a sample output:

```
OWNER       OBJECT_NAME                      VALUE
----------- ------------------------- ----------
SYSMAN      MGMT_METRICS_1HOUR_PK             19
ARUP        DLT2                              23
ARUP        DLT1                             131
```

Checking for ITL Waits

- You can check the EVENT column of V$SESSION
- The sessions will wait with the event:

```
enq: TX - allocate ITL entry
```


ITL Wait Deadlocks

An entirely different Session 3 has locked Row3 of Table2 and there is no more ITL slots on that block.

Session 1	Session 2
Update Table1 Row1	
	Update Table2 Row1
Update Table2 Row2 Hangs! Lack of ITL	
	Update Table1 Row2 *Hangs*! And deadlock

DL Graph for ITL Waits

```
Deadlock graph:
                       ---------Blocker(s)--------  ---------Waiter(s)---------
Resource Name          process session holds waits  process session holds waits
TX-0002001e-00001f21        28     116     X             26     137           S
TX-00070012-00001724        26     137     X             28     116           X

...

Rows waited on:
  Session 116: obj - rowid = 00013923 - AAATkjAAEAAAAKGAAA
  (dictionary objn - 80163, file - 4, block - 646, slot - 0)
  Session 137: no row
```

No row information, as the lock is at the block level;

The object ID here is the Table1, the victim of the deadlock; *not* the one with less ITL slots!

This is in the DL Graph

```
----- Information for the OTHER waiting sessions -----
Session 137:
  sid: 137 ser: 24985 audsid: 204469 user: 90/ARUP flags: 0x45
  pid: 26 O/S info: user: oracle, term: UNKNOWN, ospid: 6361
    image: oracle@oradba1 (TNS V1-V3)
  client details:
    O/S info: user: oracle, term: pts/2, ospid: 6360
    machine: oradba1 program: sqlplus@oradba1 (TNS V1-V3)
    application name: SQL*Plus, hash value=3669949024
  current SQL:
  update Table2 set col2 ='X' where col1 = 2;
```

The SQL gives a clue about the actual object with lack of ITL slots!

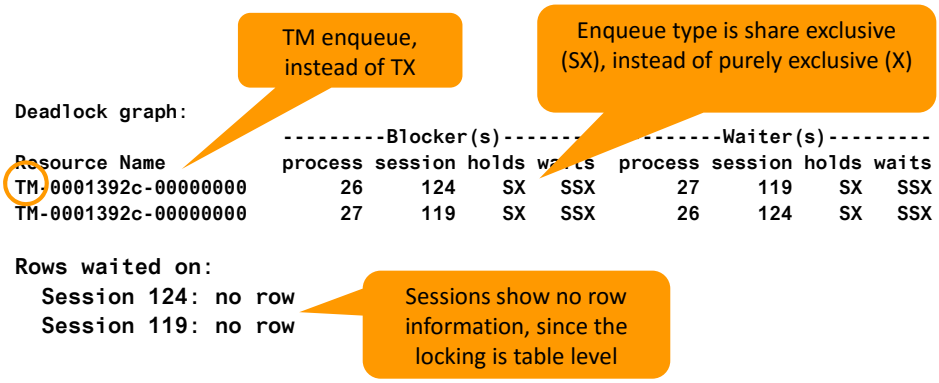
Deadlock due to Foreign Key

- When a child table row is deleted, a row share lock is taken on the parent
  - Doc is not very clear (MetaLink Bug# 2546492)
- *Unless*, an index is present
- So, this type of deadlock appears when the FK column is not indexed

FK DL

Session 1	Session 2
Delete Child Row 1	
	Delete Child Row 2
Delete Parent Row 1 Waits on TM Enqueue	
deadlock	Delete Parent Row 2 Waits for TM Enqueue

FK DL Graph



Solution: Create index on the Child on the FK column(s)

Direct Load Deadlock

- Direct Load
  - INSERT /*+ APPEND */ …
  - SQL*Loader with DIRECT=Y
- Gets a lock on the entire table

Direct Load DL

Session 1	Session 2
Direct Load into Table1	
	Direct Load into Table2
Direct Load into Table2 Waits on TM Enqueue	
	Direct Load into Table1 Waits on TM Enqueue

deadlock

DL Graph

TM lock but waits in X mode

```
Deadlock graph:
                        ---------Blocker(s)--------  ---------Waiter(s)---------
Resource Name           process session holds waits  process session holds waits
TM-0001393b-00000000        32     137     X             28     117           X
TM-0001393c-00000000        28     117     X             32     137           X

session 137: DID 0001-0020-000003BF session 117: DID 0001-001C-00001C41
session 117: DID 0001-001C-00001C41 session 137: DID 0001-0020-000003BF

Rows waited on:
  Session 137: no row
  Session 117: no row
```

No row information
Look for the SQL in the tracefile

Bitmap Index Contention

- Bitmap Index is a special type of index that stores bitmaps of actual values and compare bitmaps to bitmaps, e.g.
- Instead of "A" = "A", "01011" = "01011"
- Bitmap searches are way faster compared to literal comparison
- When a row is updated, entire bitmap index is locked, until committed

Bitmap DL

Session 1	Session 2
Update Row 1	
	Update Row 2 Hangs! for TX – Row Lock
Update Row 2	

deadlock

Bitmap DL Graph

```
Deadlock graph:
                        ---------Blocker(s)--------  ---------Waiter(s)---------
Resource Name           process session holds waits  process session holds waits
TX-00010001-00001761         28     116     X             26     117           X
TX-00050020-00001f7c         26     117     X             28     116           S

...

Rows waited on:
  Session 116: obj - rowid = 0001393E - AAATk+AAAAAAAAAAAA
  (dictionary objn - 80190, file - 0, block - 0, slot - 0)
  Session 117: obj - rowid = 0001393D - AAATk9AAEAAAAjEAAB
  (dictionary objn - 80189, file - 4, block - 2244, slot - 1)
```

Waits in Share Mode

Objects are different. 80189 is the table. 80190 is the bitmap index

PK Overlap DL

Session 1	Session 2
Insert PK Col = 1	
	Insert PK Col = 2
Insert PK Col = 2 Hangs with "S"	
	Insert PK Col = 1 Hangs and deadlock

PK Overlap Graph

```
Deadlock graph:
                         ---------Blocker(s)--------  ---------Waiter(s)---------
Resource Name            process session holds waits  process session holds waits
TX-00080011-00001f0b         26     137     X             28     116           S
TX-0003000e-00001f5c         28     116     X             26     137           S

...

Rows waited on:
  Session 137: no row
  Session 116: no row
```

No row information; so difficult to diagnose immediately. Look for the SQL statement.

Special Cases

- Some interesting cases I have encountered

Autonomous Txns

- Tracefile shows only one session. The other session information is not even there

```
Deadlock graph:
                    ---------Blocker(s)-------- ---------Waiter(s)---------
Resource Name       process session holds waits process session holds waits
TX-0005002d-00001a40 17      14          X            17      14           X

session 14: DID 0001-0011-00000077
session 14: DID 0001-0011-00000077
Rows waited on:
Session 14: obj - rowid = 000078D5 - AAAHjVAAHAAAACOAAA
(dictionary objn - 30933, file - 7, block - 142, slot - 0)
Information on the OTHER waiting sessions:
End of information on OTHER waiting sessions.
```


PQ Deadlocks

- Procedural Logic

```
LOOP
   SELECT /*+ PARALLEL */ FOR UPDATE
END LOOP
```

- Kicked off more than once concurrently
- Update the same rows from many sessions, PQ slaves do not know
- Deadlocks among the PQ slaves

Other Cases

- Triggers Firing Autonomous Transaction
- Freelists – If too many process freelists are defined, it's possible to run out of transaction freelists

In Conclusion

- Most common cause of deadlocks is the common locking
- But that's not the only reason
- ITL Shortage, Bitmap Index Locking, Lack of FK Index, Direct Path Load, PK Overlap are also causes.
- Check the tracefile to determine the possible causes

Summary

- Latches are just memory structures in SGA
- Provide a locking mechanism for buffer headers, library cache objects, etc.
- No queueing. First come first serve
- X$ and V$ views show the latch activity
- If you see a latch contention,
  - Buffer latch: too much buffer access
  - Shared pool latch: too much concurrent access to objects

Thank You!

Blog: arup.blogspot.com
Tweeter: @ArupNanda
Facebook.com/ArupKNanda
Google Plus: +ArupNanda