Oracle Diagnostics

Hemant K Chitale

Hemant K Chitale

- whoami ?
- Oracle 5 to Oracle 10gR2 : DOS, Xenix,8 flavours of Unix, Linux, Windows
- Financial Services, Govt/Not-for-Profit, ERP, Custom
- Production Support, Consulting, Development
- A DBA, not a Developer
- Product Specialist, Standard Chartered Bank
- *My Oracle Blog* *http://hemantoracledba.blogspot.com*

Locks and Lock Trees

- Row Locks are Enqueues
- They serialise access to rows
- A transaction may hold Row Locks on multiple rows – this is represented as a single entry in V$TRANSACTION but single or multiple entries in the ITL slots in various table / index blocks
- ITLs allow different transactions to lock different rows in the same block concurrently.
- Lock Trees are multiple sessions waiting “in order”, with potentially more than one session waiting on the same row lock

A Lock Tree :

Script “utllockt.sql” (in $ORACLE_HOME/rdbms/admin) can provide a tree-like diagram.

```
 * This script  prints  the  sessions in   the system  that  are waiting for
 * locks,  and the locks that they  are waiting for.   The  printout is tree
 * structured.  If a sessionid is printed immediately below and to the right
 * of another session, then it is waiting for that session.  The session ids
 * printed at the left hand side of the page are  the ones  that everyone is
 * waiting for.
 *
 * For example, in the following printout session 9 is waiting for
 * session 8, 7 is waiting for 9, and 10 is waiting for 9.
 *
 * WAITING_SESSION   TYPE MODE REQUESTED    MODE HELD         LOCK ID1 LOCK ID2
 * ----------------- ---- ----------------- ----------------- -------- --------
 * 8                 NONE None              None              0         0
 *    9              TX   Share (S)         Exclusive (X)     65547     16
 *       7           RW   Exclusive (X)     S/Row-X (SSX)     33554440  2
 *       10          RW   Exclusive (X)     S/Row-X (SSX)     33554440  2
 *
 * The lock information to the right of the session id describes the lock
 * that the session is waiting for (not the lock it is holding).
```

The script can be enhanced to provide more session information. The script uses DDLs to drop and create temp tables – so another enhancement would be to have those tables created in advance as GTTs and only populated and queried by the script

Here is a query to list Lock Holders and Waiters:

```
select s.blocking_session, to_number(s.sid) Waiting_Session, s.event, s.seconds_in_wait,
p.pid,
       p.spid "ServerPID", s.process "ClientPID",
       s.username, s.program, s.machine, s.osuser, s.sql_id,
       substr(sq.sql_text,1,75) SQL
      from v$sql sq, v$session s, v$process p
where s.event like 'enq: TX%'
and s.paddr=p.addr
and s.sql_address=sq.address
and s.sql_hash_value=sq.hash_value
and s.sql_id=sq.sql_id
and s.sql_child_number=sq.child_number
union all
select s.blocking_session, to_number(s.sid) Waiting_Session, s.event, s.seconds_in_wait,
p.pid,
       p.spid "ServerPID", s.process "ClientPID",
       s.username, s.program, s.machine, s.osuser, s.sql_id,
       substr(sq.sql_text,1,75) SQL
      from v$sql sq, v$session s, v$process p
where s.sid in (select distinct blocking_session from v$session where event like 'enq:
TX%')
and s.paddr=p.addr
and s.sql_address=sq.address(+)
and s.sql_hash_value=sq.hash_value(+)
and s.sql_id=sq.sql_id(+)
and s.sql_child_number=sq.child_number(+)
order by 1 nulls first, 2
/
```

This method does NOT require any temporary tables !

Example 1 :

Two separate sessions attempting to update the same row :

```
SQL> connect  ABC_DBA/ABC_DBA_123
Connected.
SQL> update hemant.test_row_lock set content = 'Another' where
pk=1;

1 row updated.

SQL>
```

```
SQL> connect hemant/hemant
Connected.
SQL> update test_row_lock set content = 'First' where pk=1;
```

….. now waiting …..

```
BLOCKING_SESSION WAITING_SESSION EVENT
SECONDS_IN_WAIT     PID
---------------- --------------- ----------------------------------------------------------------
-------- -------------- ----------
ServerPID                ClientPID                USERNAME                       PROGRAM
------------------------ ------------------------ ------------------------------ --------
----------------------------------------
MACHINE                                                          OSUSER
SQL_ID
---------------------------------------------------------------- ------------------------
------ -------------
SQL
----------------------------------------------------------------------------------------
-------------------------------------------
                              17 SQL*Net message from client
82      27
13788                    13770                    ABC_DBA
sqlplus@localhost.localdomain (TNS V1-V3)
localhost.localdomain                                            oracle


              17              26 enq: TX - row lock contention
52      19
13791                    3449                     HEMANT
sqlplus@localhost.localdomain (TNS V1-V3)
localhost.localdomain                                            oracle
fuwn3bnuh2axg
update test_row_lock set content = 'First' where pk=1


SQL>
```

Findings for Ex. 1 :

Session 17 (ABC_DBA) has no ‘BLOCKING_SESSION’. It isn’t waiting. It is the Blocker itself.

Session 26 (HEMANT) as a ‘WAITING_SESSION’ presents Session 17 as being the ‘BLOCKING_SESSION’.

Session 26 is waiting on a Row Lock while currently running “`update test_row_lock set content = 'First' where pk=1`”

Example 2 :

Table Definition :

```
SQL> create table test_unique_insert_row_lock
(col_1 number , col_2 varchar2(5));

Table created.

SQL> create unique index t_u_i_r_l_unq_ndx on
test_unique_insert_row_lock (col_1);

Index created.

SQL>
```

Example 2 (contd)**:**

```
SQL> connect ABC_DBA/ABC_DBA_123
Connected.
SQL> insert into hemant.test_unique_insert_row_lock values
(1, 'Sn.1');

1 row created.

SQL>
SQL> connect hemant/hemant
Connected.
SQL>  insert into hemant.test_unique_insert_row_lock values
(1,'Sn.2');
```
….. now waiting …..

```
BLOCKING_SESSION WAITING_SESSION EVENT
SECONDS_IN_WAIT PID
---------------- --------------- --------------------------------------------------
-------------- --------------- ----------
ServerPID                ClientPID                USERNAME
PROGRAM
------------------------ ------------------------ ------------------------------ --
----------------------------------------------
MACHINE                                                           OSUSER
SQL_ID
---------------------------------------------------------------- ------------------
------------ -------------
SQL
----------------------------------------------------------------------------
                               1 SQL*Net message from client
115  22
13007                    12813                    ABC_DBA
sqlplus@localhost.localdomain (TNS V1-V3)
localhost.localdomain                                             oracle


               1              36 enq: TX - row lock contention
80  25
13009                    12971                    HEMANT
sqlplus@localhost.localdomain (TNS V1-V3)
localhost.localdomain                                             oracle
166dv336yhxua
 insert into hemant.test_unique_insert_row_lock values (1,'Sn.2')


2 rows selected.

SQL>
```

Findings for Ex 2 :

Session 1 (ABC_DBA) has no blocker

Session 36 (HEMANT) is waiting on Session 1

So, even a Duplicate Value on INSERT actually “waits” on the INSERT that has not committed. The row inserted by ABC_DBA is not visible to HEMANT (because ABC_DBA has not yet committed) but HEMANT’s INSERT waits on the Row Lock !