Enqueue and locks are complex structure in Oracle. This book clear all the concept about it.

Enqueue and Locks

Enqueue, DML locks, Case Studies

Techgoeasy

1 Enqueue and Locks

Topics Covered

1. What is Enqueue and locks
2. What is Enqueue identifier?
3. What is Enqueue resource?
4. How is the lookup done in Resource table?
5. How is Enqueue operations works?
6. How is the queue checked when lock is release or converted
7. Common Types of Enqueue
8. Views and Table to view Enqueue and locks
9. How the DML locks are handled in Oracle server
10. How Table locks are Implemented
11. How to disable Table locks
12. How DML row locks are implemented
13. What is interested transaction list
14. What is transaction
15. Detailed Example of DML locks

What is Enqueue and locks

Enqueue are locks which serialize the operations to the shared structure. The shared structure could be table, redo threads and transactions.

When a user A updates a row 12 in the table, it acquires the transaction Enqueue (lock). This is acquired so that Any user which I trying to update that same row 12 in the table will wait until the user A commit the transaction. So now if user B tries to update the same row, it will wait on the enqueue.

Once the user A commits the transaction, User B transaction will proceed

We have local enqueue in the single instance database while with RAC, we have local enqueue and global enqueue to manage the shared resource

What is Enqueue identifier

Enqueue are uniquely identified using the format

<Resource type>_<id1>_<id2>

Resource can

TM -> table locks

MR-> Media recovery

TX-> Transaction

Id1 and id2 are number which are different for different resource type

Like for table lock (TM), it is wriiten as

TM-<object id of the table>-0

When a user request access to the resource in certain mode, an enqueue identifier is generated which explained above

Enqueue are held in these mode

SS: Row share mode

SX: Row exclusive mode

S: Lock the table in share mode

SSX: Lock the table in share mode and row in exclusive mode

X: Lock the table in exclusive mode

What is Enqueue resource

Each enqueue is maintained through a resource structure by Oracle server and it is identified as explained above. A resource structure has three lists

a) Owner list

b) Waiting list

c) Converter list

When a user requests a lock on resource in certain mode, it obtained a lock structure and it makes a request to acquire the lock on certain resource.it is the placed in these lists of the resource structure as per the lock required.

So the user is first requesting that resource, then it will be placed in owner list

Resource structure are obtained from Resource table and lock structure are obtained from Lock table .They are both allocated in SGA

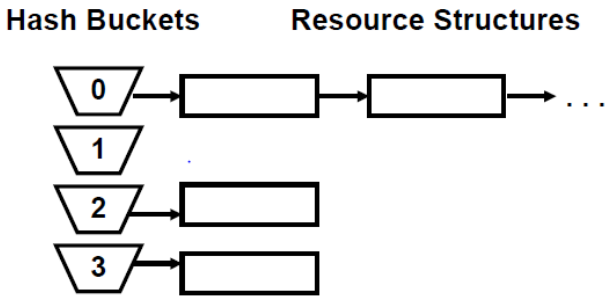
The number of rows in resource table is defined by the initialization parameter enqueue_resources.The values in use can be seen in v$resource view

The number of rows in lock structure table is defined by the initialization parameter _enqueue_locks. The values in use can be seen in v$enqueue_lock

How is the lookup done in Resource table?

1) The resource table contains all the resource structure. A hashing algorithm is used to find and access the resource structure in the resource table.

2) The resource table is arranged in hash bucket. Each hash bucket contains a list of resource structure in linked list form.



3) When the resource is search, its hash is obtained using hashing algorithm and then the latch is obtained to find the corresponding hash bucket and then the resource is search on the list in hash bucket. If the resource is found, Lock structure is obtained and the request is placed on owner, waiter and convert list as per the specified level of lock requested

   Example TM-575-0 resource hashed to bucket 1, A latch enqueue hash chain is obtained to access the hash bucket and the list is accessed in the bucket to obtain the resource structure

4) If the resource is not found in bucket list and a new resource structure is obtained from resource free list and placed in the bucket list. This happens under the latch Enqueue. A lock structure is also allocated

   The lock request is placed on the owner list of the resource structure

How is enqueue operations works?

When a user request a lock on the resource, Oracle server does following things

1) If it is not currently owned, the resource is granted to the user
2) If it is owned and there are waiters and converter, then it is placed at the bottom of waiters queue
3) If it is owned but there are no waiter and converter then if it is compatible with the owner lock, the request is granted. If it is not compatible, then it is placed on waiter list
4) A converter is allowed to proceed if the request is less restrictive than the lock held currently or requested mode is compatible with the lock held by the other owner
5) A waiter is allowed to proceed if the converter list is empty , there is no waiters ahead of it and lock requested is compatible with the lock currently it has
6) Converter is always processed before the waiters.
7) Oracle server checks these queues every time lock is released or converted.

How the queue is checked when lock is release or converted

1) Processes waiting for the resources sleep on the semaphores, and semaphores are used as sleep/wakeup mechanisms. After enqueueing in to the queue, the requesting process will sleep on the semaphore using the sync_op call.

sync_op(SYNC_WAIT, SYNCF_BINARY, 300) = 1

2) Once the process holding the resource is ready to release the resource, it looks at the queue attached to the resource structure. If there's a process in the queue, it sends a semaphore signal to the waiting process using

sync_op call.

sync_op(0x0005, SYNCF_BINARY, 134491620) = 1

3) The waiting process will handle the signal and will wake up. This waiting process modify the status as per the steps given in enqueue operation

Common Types of enqueue

JQ - Job Queue. When a job (submitted by DBMS_JOB.SUBMIT) is running, it is protected by a JQ enqueue (which means that only one SNP-process can run the job).

ST - Space management Transaction. The ST enqueue is need to be held every time the session is allocating/deallocating extents (which means wants to change the UET$ and FET$ dictionary tables), like coalescing, drop/truncate segments and disk -sorting. If the session gets a timeout when requesting the ST enqueue, "ORA-1575 timeout waiting for space management" is returned.

TM - DML (Table) enqueue. Every time a session wants to lock a table, a TM enqueue is requested. If a session deletes a row in the parent-table (DEPT) and a referential constraint (foreign key) is created without an index on the

child-table (EMP), or if the session is updating the column(s) that the foreign key references to then a share lock (level 4) is taken on the child table. If another session tries to do changes to the child-table they have to wait (because they want the enqueue in row exclusive mode, and that is not compatible with the share mode). If an index is created on the child-table's foreign key-column, then no share- lock is required on the child-table.

TX - Transaction. As soon as a transaction is started a TX enqueue is needed. A transaction is uniquely defined by the rollback segment number, the slot number in the rollback segment's transaction table and the slot number's sequence number. A session can be waiting on a TX enqueue for several reasons:
1) Another session is locking the requested row.
2) When two sessions tries to insert the same unique key into a table (none of them has done a COMMIT), then the last session is waiting for the first one to COMMIT or ROLLBACK.
3) There are no free ITL (Interested Transaction List) in the block header (increase INI_TRANS och PCT_FREE for the segment).

UL - User Lock. A session has taken a lock with the DBMS_LOCK.REQUEST Function.

Views and Table to view enqueue and locks

a) V$session and v$session_wait

When is session is waiting on enqueue or lock, this can be session from V$session (in 11g and above) and v$session_wait

We can use below query to obtain all the enqueue in the system

```
Select * from v$session_wait where event like 'enq%';

The parameter of the enqueue wait event has following meaning

P1: resource type and mode wanted

P2:ID1 of the resource

P3: ID2 of the resource
```

```
Select event,p1, p2,p3 from v$session_wait  where wait_time=0 and event like 'enq%';
```

b) V$lock is another useful view to check enqueue 's

i) V$lock list all the lock structure currently held in the system

ii) The column type ,id1 and id2 represent the resource type ,id1 and id2 of the resource structure.so it can be joined with V$resource which contains the list of all the resource structure

iii) LMODE and request tells us which queue (owner,converter,waiters) is the session

LMODE	Request	Queue name
> 0	=0	Owner
=0	> 0	Waiter
> 0	> 0	Converter

Below query can be used to find holder and waiter

```
SELECT inst_id,DECODE(request,0,'Holder: ','Waiter: ')||sid sess,
        id1, id2, lmode, request, type
    FROM V$LOCK
    WHERE (id1, id2, type) IN
        (SELECT id1, id2, type FROM V$LOCK WHERE request>0)
    ORDER BY id1, request
   ;
```

In case of RAC

```
SELECT inst_id,DECODE(request,0,'Holder: ','Waiter: ')||sid sess,
        id1, id2, lmode, request, type
    FROM GV$LOCK
```

```
WHERE (id1, id2, type) IN
      (SELECT id1, id2, type FROM gV$LOCK WHERE request>0)
ORDER BY id1, request
;
```

c) V$locked_object is another useful view

It contains all the TM locks in the database. It gives the transaction slot, OS process is and session id of the session which is holding the TM locks

d) There are several views which can be used to find the locks information. These views are created by catblock.sql

DBA_LOCKS	Show all the locks like v$lock
DBA_DML_LOCKS	Shows all DML™ locks held or being requested
DBA_DDL_LOCKS	Shows all DDL locks held or being requested
DBA_WAITERS	Shows all sessions waiting on, but not holding waited for locks
DBA_BLOCKERS	Shows non-waiting sessions holding locks being waited-on

Query to find out waiting session and holding sessions

```
set linesize 1000
```

```
column waiting_session heading 'WAITING|SESSION'
column holding_session heading 'HOLDING|SESSION'
column lock_type format a15
column mode_held format a15
column mode_requested format a15

select
  waiting_session,
  holding_session,
  lock_type,
  mode_held,
  mode_requested,
  lock_id1,
  lock_id2
from
  dba_waiters
/
```

Query to find out all the locked objects

```
set term on;
set lines 130;
column sid_ser format a12 heading 'session,|serial#';
column username format a12 heading 'os user/|db user';
column process format a9 heading 'os|process';
column spid format a7 heading 'trace|number';
column owner_object format a35 heading 'owner.object';
column locked_mode format a13 heading 'locked|mode';
column status format a8 heading 'status';
select
   substr(to_char(l.session_id)||','||to_char(s.serial#),1,12) sid_ser,
   substr(l.os_user_name||'/'||l.oracle_username,1,12) username,
   l.process,
   p.spid,
   substr(o.owner||'.'||o.object_name,1,35) owner_object,
   decode(l.locked_mode,
          1,'No Lock',
          2,'Row Share',
          3,'Row Exclusive',
          4,'Share',
```

```
          5,'Share Row Excl',
          6,'Exclusive',null) locked_mode,
   substr(s.status,1,8) status
from
   v$locked_object l,
   all_objects     o,
   v$session       s,
   v$process       p
where
   l.object_id = o.object_id
and l.session_id = s.sid
and s.paddr      = p.addr
and s.status != 'KILLED'
/
```

How the DML locks are handled in Oracle server

When an update, inserts, deletes or select for update is executed on the table, Oracle takes these two locks

1) DML Table Lock: To ensure object definition consistency for the duration of the transaction. This prevents any DDL operations to occur while a DML is in progress.
2) DML Row Lock: This is to ensure consistency of the data during the execution of the transaction. We can rephrase like This obtains a lock on the particular row being touched and any other transaction attempting to modify the same row gets blocked, till the one already owning it finishes

How table locks are Implemented

We already explained the infrastructure of Enqueue in previous chapter. Table locks are implemented as TM Enqueue

So Enqueue structure would be
TM-<Table object id> -0

Modes are
RS: row share
RX: row exclusive
S: share
SRX: share row exclusive
X: exclusive

Each cursor maintains a list of table lock structure which is built while parsing the statement. Upon the first execution, function call is made to lock all the table in the list. The locks are released when the transaction commits or rollback.

The possibility of rollback, particularly rollback to a save point, adds another dimension of complexity to dictionary locking. Namely, if a transaction is rolled back beyond the point at which a lock was upgraded, then the lock must be downgraded correspondingly, as part of the rollback operation, in order to reduce the risk of artificial deadlocks.

The requirements of dictionary locking for transactions and, in particular, the maintenance of a history of lock conversions, are provided by DML locks in conjunction with TM enqueue. Every transaction holding a DML lock also holds a TM enqueue lock. The basic locking functionality is provided by the enqueue, and the DML lock adds the maintenance of the conversion history.

The fixed array of DML lock structures is sized by the DML_LOCKS parameter. Its free list is protected by the dml lock allocation latch, and the active slots are visible in V$LOCKED_OBJECT .

To set the DML_LOCKs check the utilization in v$resource_limit. We can set it generously as it takes very less space

How to disable the table locks?

1) DML locks and the associated TM enqueue locks can be disabled, either entirely, or just for certain tables.

2) To disable these locks entirely, the DML_LOCKS parameter must be set to zero. In a parallel server database, it must be set to zero in all instances.
3) To disable such locks against a particular table, the DISABLE TABLE LOCKS clause of the ALTER TABLE statement must be used.
4) If locks are disabled for a table, then DML statements can still modify the table's blocks, and row-level locks are still held. However, the sub-shared mode table locks normally associated with queries, and the sub-exclusive mode table locks normally associated with DML, are not taken. Instead, transactions against the table are protected from conflicting DDL by simply prohibiting all attempts to take a lock on the entire table, and thus all DDL against the table.
5) Disabling the table locks can increase performance as lock acquisition overhead is reduce ed. This is more particularly important in case of RAC where this overhead is quite high.
6) Disabling the table locks also prevent creating foreign key indexes. As foreign key need to indexed in order to avoid table lock of the child table while the rows are manipulated in parent table. So if we disable table lock all together then indexes are not required
7) It is preferable to use alter table to disable the table locks on some table then to set to dml_locks table. As if dml_locks is set to zero, we will need to bounce the instance to set it again
8) In direct load insert, a session will take the TM enqueue in 'X' mode. This prevents any other DML from taking place while the direct load is happening, in addition to blocking all DDL

How DML row locks are implemented

DML Row locks are implemented as combination of following two things

a) Row level lock: It is implemented as lock byte in each row header and ITL(interested transaction list) in each data or index block. These are not cached anywhere and since they are stored in block itself not in SGA which is limited, this mechanism of Locking by oracle is massively scalable

b) Transaction locks: These are implemented as TX Enqueue

The lock byte points to the ITL entry in the block and All ITL entries for the transaction points to the TX Enqueue which ultimately determines if the transaction is committed or rollback. The waiters will wait on transaction lock

Example

1) A transaction A wants to update rows 2 and 3 in the block. It will allocate a ITL(interested transaction list). The transaction accesses the row 2 and 3 and sees the lock byte .If the lock byte is zero, It is not locked. The transaction will update the row 3 ,3
2) Now a transaction B start and it wants to update the rows 1 . It will allocate a ITL(interested transaction list). The transaction accesses the row 1 and sees the lock byte .If the lock byte is zero, It is not locked. The transaction will update the row 1
3) Now the transaction wants to update the row 2. It will access the row and will find it lock as lock byte will not be zero. It will look in the ITL which holds the lock. It will perform ITL cleanout to find out if the transaction is active or not active. In this case ,it will find Transaction A active. So transaction B has to wait on transaction A to rollback or commit. The transaction B will wait on demanding the TX Enqueue which the transaction A hold in exclusive mode

What is Interested Transaction list

When a session wants to modify a block, it has to allocate a ITL in the block. ITL is the data structure in the block header which contains many slots which are taken by the transaction. It is defined by the parameter INITRANS and MAXTRANS when the table is created. Initial numbers of slots are created as per INITTRANS and they grow dynamically to the maximum of MAXTRANS

What is transaction?

When a session update /delete/insert ,then a transaction is started. It is completed when the commit or rollback happened. A transaction is identified by transaction identifier(XID). The transaction identifies consists of three part

a) Rollback or undo segment number
b) Transaction table Slot number
c) Sequence or wrap no

XID= usn#.slot#.wrap#

Each block ITL will contain the XID

A ITL cleanout means to look for the XID in the ITL and search the rollback segments based on this and look the for the transaction table and wrap number to check for the transaction activeness.

We can use below command to dump any rollback segment
Alter system dump undo header <undo segment name>;

Each active transaction can be in seen in v$transaction table

```
select addr, xidusn, xidslot, xidsqn
  from v$transaction;

ADDR        XIDUSN   XIDSLOT    XIDSQN
-------- ---------- ---------- ----------
3C485875         50          5       3000
```

The transaction Identifier (XID) can also be obtained in the own session using

```
select dbms_transaction.local_transaction_id from dual;
```

The wait on TX enq will be seen in v$session_wait
P1:Name|mode
P2:rbs3|wrap#
P3:slot#

To summarize the DML row locks

The first DML in a session where a transaction does not already exist will implicitly create a transaction.

- An undo segment number, slot, and wrap will be assigned
- TX enqueue will be instantiated

When a row to be modified is identified, session will take an entry in the ITL of the data block, assign it to the transaction

- USN/SLOT/WRAP will be written to ITL slot, reserving that slot for the current transaction
- Lock will be taken on the row, by setting the lock byte in the row directory to point to the current transaction's ITL slot

Both the TM and TX Enqueue can be seen in V$lock

1) Type identifies TM or TX
2) ID1 and ID2 may carry additional information, but are context-sensitive with respect to the enqueue TYPE
3) For TM enqueue, ID1 is the OBJECT_ID of the object being locked, which can be referenced in DBA_OBJECTS, and ID2 is always 0
4) For TX Enqueue, ID1 and ID2 hold the undo segment number, slot number, and wrap

Detailed Example to explain the DML locks

1) Create the dummy table

Create table from j as select * from dba_objects where rownum < 3;

Table created

Create table from j1 as select * from dba_objects where rownum < 3;

Table created

2) Session A

Select * from j for update;

Let's see what is present in v$lock

Select distinct sid from v$mystat

SQL> select distinct sid from v$mystat;

```
       SID
----------
      2125
```

SQL> select * from v$lock where sid=2125;

```
ADDR             KADDR                SID TY       ID1       ID2     LMODE
---------------- ---------------- ---------- -- ---------- ---------- ----------
  REQUEST     CTIME     BLOCK
---------- ---------- ----------

00000006B5D9D0D0 00000006B5D9D148       2125 TX    2883613   16425600          6
         0         44          0

FFFFFFFF7DA4B360 FFFFFFFF7DA4B3C0       2125 TM   21488781          0          3
         0         44          0
```

So we see here

1) DML table lock is created
2) TX(transaction lock) is created

3) Let's start session B

Select * from j1 for update;

SQL> select distinct sid from v$mystat;

```
       SID
----------
      2302
```

```
SQL> select * from v$lock where sid=2302;

ADDR             KADDR                SID TY        ID1        ID2      LMODE
---------------- ---------------- ---------- -- ---------- ---------- ----------
  REQUEST      CTIME      BLOCK
---------- ---------- ----------
00000006AF7FF910 00000006AF7FF988       2302 TX    2949148   16884039          6
         0         10          0

FFFFFFFF7DA4B360 FFFFFFFF7DA4B3C0       2302 TM      33544          0          3
         0         10          0

00000006DC289D60 00000006DC289DB8       2302 AE   15062272          0          4
         0        106          0
```

So we see here
4) DML table lock is created
5) TX(transaction lock) is created

Now let's try to do

Select * from j for update;

This will hang

3) Let start the another session to analyze the issue

If you see the session sid =2032 details in V$lock

```
SQL> select * from v$lock where sid=2302;

ADDR             KADDR                 SID TY       ID1       ID2      LMODE
---------------- ---------------- ---------- -- ---------- ---------- ----------
  REQUEST     CTIME      BLOCK
---------- ---------- ----------

FFFFFFFF7DA4B360 FFFFFFFF7DA4B3C0      2302 TM      33544          0         3
         0         47          0

00000006DC289D60 00000006DC289DB8      2302 AE    15062272          0         4
         0        143          0

00000006DC289808 00000006DC289860      2302 TX     2883613   16425600         0
         6          7          0

FFFFFFFF7DA4B360 FFFFFFFF7DA4B3C0      2302 TM    21488781          0         3
         0          7          0
```

The bolded row is request 6(exclusive lock) on some TX enq

Now we can use below query to find the blocking session

```
select l1.sid, ' IS BLOCKING ', l2.sid
    from v$lock l1, v$lock l2
    where l1.block =1 and l2.request > 0
    and l1.id1=l2.id1
    and l1.id2=l2.id2

     SID 'ISBLOCKING'         SID
---------- ------------ ----------
     2125  IS BLOCKING        2302
```

Now we can commit or rollback the session 2125 in order for the transaction B to proceed. We can kill the session 2125 using the below command also to release the lock

Alter system kill session ‘2125,<serial>’;

Some more additional information

The TX locks in v$lock does not tell the row information where the contention is present. We can view those things using the queries

The below query need to be executed from the session which is waiting

```
SQL> select row_wait_obj#, row_wait_file#, row_wait_block#, row_wait_row#
  from v$session where sid=2302

ROW_WAIT_OBJ# ROW_WAIT_FILE# ROW_WAIT_BLOCK# ROW_WAIT_ROW#
------------- -------------- --------------- -------------
    21488781            461           81063             0

SQL> select do.object_name,
    row_wait_obj#, row_wait_file#, row_wait_block#, row_wait_row#,
    dbms_rowid.rowid_create ( 1, ROW_WAIT_OBJ#, ROW_WAIT_FILE#, ROW_WAIT_BLOCK#, ROW_WAIT_ROW# )
    from v$session s, dba_objects do
    where sid=2302
    and s.ROW_WAIT_OBJ# = do.OBJECT_ID ;  2    3    4    5    6

OBJECT_NAME
--------------------------------------------------------------------------------
ROW_WAIT_OBJ# ROW_WAIT_FILE# ROW_WAIT_BLOCK# ROW_WAIT_ROW# DBMS_ROWID.ROWID_C
------------- -------------- --------------- ------------- ------------------
J
    21488781            461           81063             0 ABR+SNAHNAAATynAAA



Select * from j where rowid=’ ABR+SNAHNAAATynAAA’;
```