

# Latches Demystified

Arup Nanda  
*Longtime Oracle DBA*

## What is a "Latch"

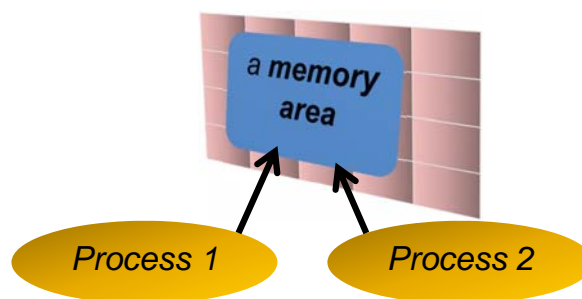
From "Glossary" in Oracle Manuals:

“*A low-level serialization control mechanism used to protect shared data structures ...*”

# Agenda

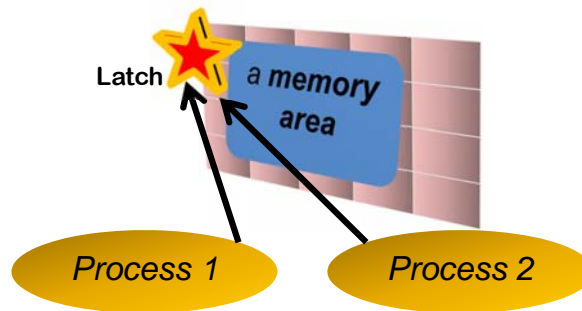
- What are latches – the purpose
- Buffer cache latches
- Shared pool latches
- Identifying latch waits
- When the database is hung
- Plenty of demos

# Latches



**If process 1 and 2 both go after the memory area at the same time, they will end up corrupting the area. Who makes sure they get their turns?**

# Latches



- 👉 Process 1 and 2 will try to get the “latch”, a area in memory that does not have any required data.
- 👉 Whoever gets the latch now gets to access the memory area exclusively
- 👉 When done, the process releases the latch

# Spinning and Sleeping

- Suppose process 1 gets the latch, accesses the memory
- How will process 2 know when the latch is available?
  - No central latch repository
  - No communication to the process
- Process 2 will constantly loop to check if the latch is free
- This is called **spinning** – a CPU intensive process
- After  $n$  times, it will stop spinning and will go to sleep
  - $n = \_spin\_count$  in init.ora, defaults to 2000
- After that it will wake up after 1 ms, check, go to sleep
- Check again in 1ms, sleep, then check in 2 ms, sleep ...

## Latches

- 100 or 200 bytes memory in SGA (depending on 32 or 64 bit Oracle)
- Value depends on how it has been taken



Untaken



Exclusive



Sharable;  
but taken  
exclusively



Sharable;  
taken by  
many  
processes

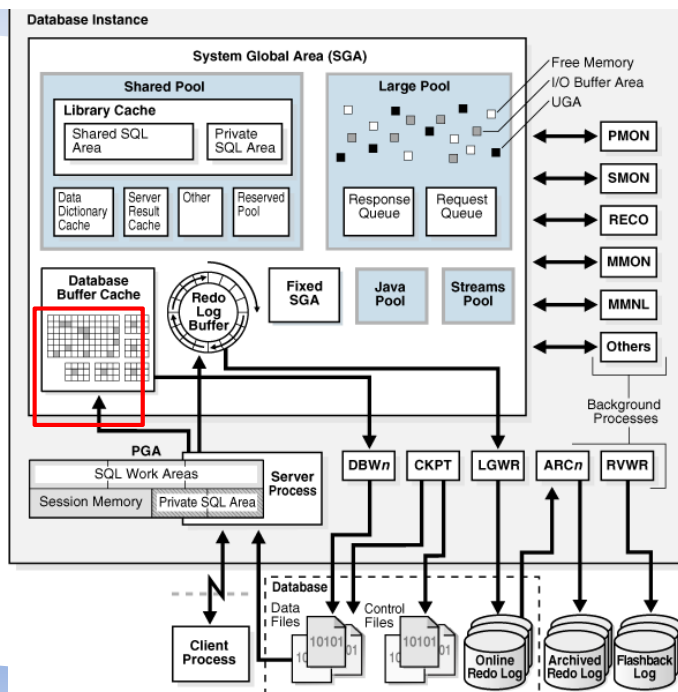
## Information on Latches

- V\$LATCH – latch
- V\$LATCH\_CHILDREN – the child latches
- V\$LATCH\_PARENT – the parent latches
- V\$LATCHHOLDER – the holder of latches
  - PID – the process ID
  - SID – the session SID
  - LADDR – the address of the latch
  - NAME – name of the latch
  - GETS – how many times it got the latch

# Latches -vs- Locks

Latches	Locks
On physical components like memory and CPU	On logical structures like rows
No queues	Queues
No ordering	No ordering
When multiple processes compete for the same resource; no guarantee on which one gets it	The sessions get the lock in the order they wait

# Oracle Instance

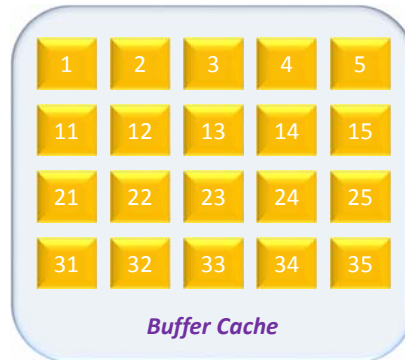


Source: Oracle Database Documentation Concepts Guide

# Buffer Operation



SELECT ...  
FROM EMP  
WHERE ...

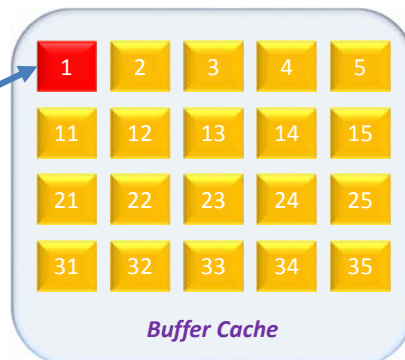


# Buffer Operation

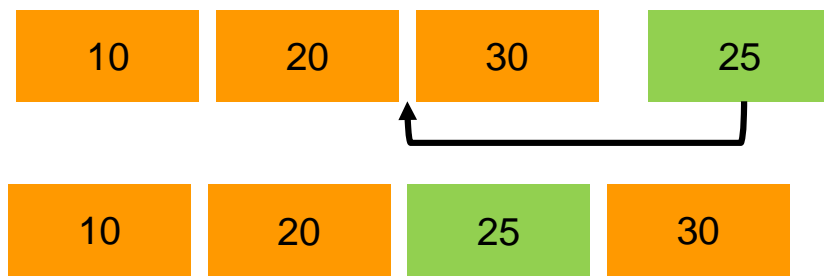


Data Block

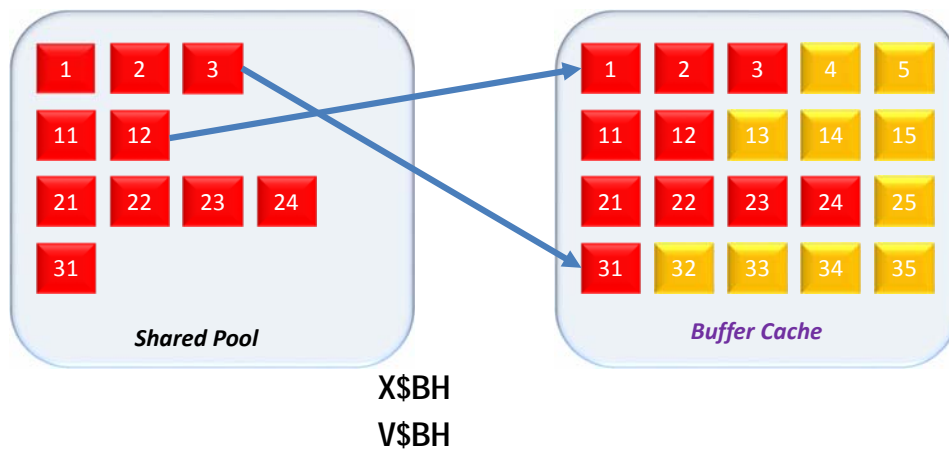
SELECT ...  
FROM EMP  
WHERE ...



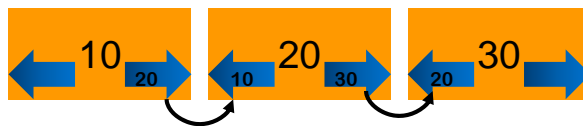
# Buffer Insertion



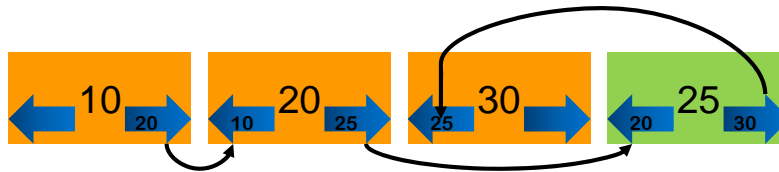
# Buffer Header



# Buffer Header Management

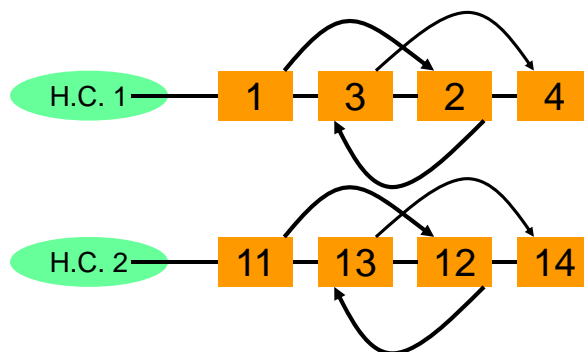


Linked List



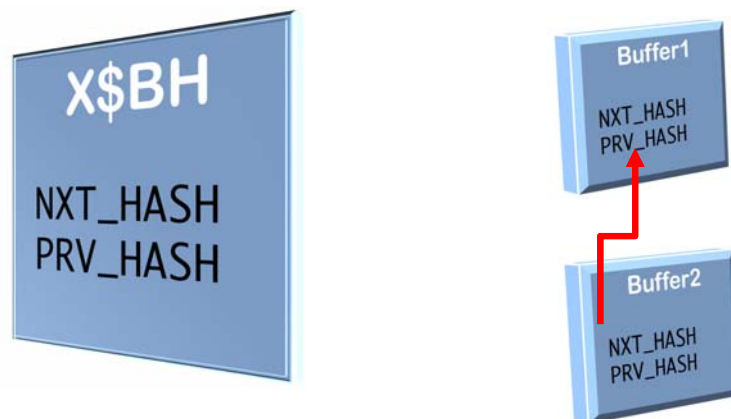
When a new buffer comes in, only the pointers are updated

# Buffer Cache





## Linked List

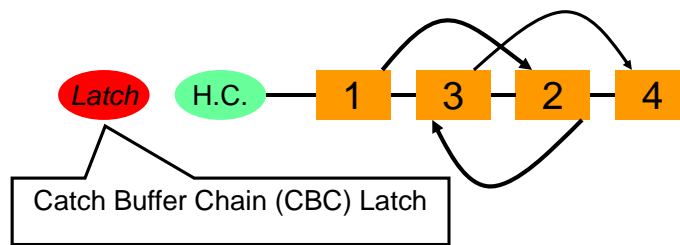


## Test for Buffer Header

```
select
  ltrim(addr,'0') buffer_address,
  ltrim(nxt_hash,'0') next_buffer,
  ltrim(prv_hash,'0') prev_buffer,
  case
    when nxt_hash = prv_hash then 'Unlinked'
  else
    'Linked'
  end
  as linked
from x$bh
where hladdr = '000007FF3C8B1568'
```

bh1.sql

## Buffer Cache



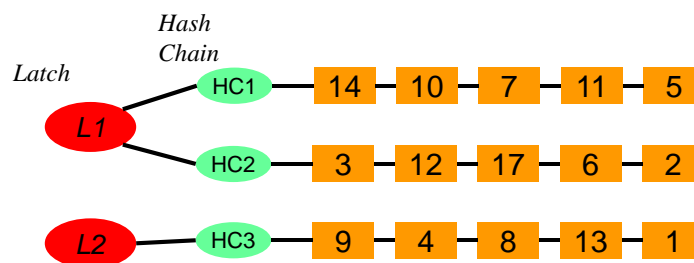
## Identifying Buffer Latches

- Demo
- Find out the rows and blocks – `qsales.sql`
- Find out the data object id – `dobjid.sql`
- Find out the data block address – `dba1.sql`
- Find out the child latch address – `hladdr1.sql`
- Find out the partition name – `extents1.sql`
- Find out the objects protected by a latch – `latchobjs.sql`
- Find out the total buffers per latch – `clatchcount.sql`

## Identifying CBC Latch Contention

- EVENT column in V\$SESSION shows '%cache buffer%'
- Also in V\$ACTIVE\_SESSION\_HISTORY
- Find out the history – ash latch.sql
- Convert to hex – tohex.sql
  
- Blog entry <http://arup.blogspot.com/2014/11/cache-buffer-chains-demystified.html>

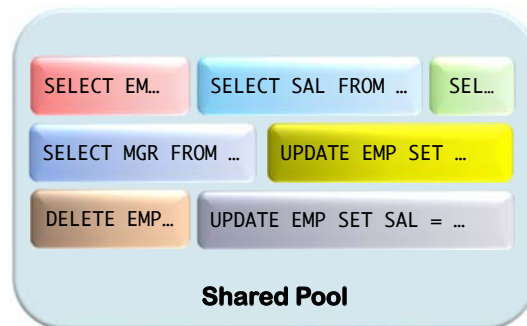
## Latches and Hash Chains



No. of hash buckets = init.ora parameter `_db_block_hash_buckets`  
No. of latches = `_db_block_hash_latches`

# Library Cache Latches

SELECT ENAME FROM EMP



# Library Cache Latch Modes



PIN



LOCK

## Check Library Cache

```
select
  s.sid,
  ob.kglnaown obj_owner,
  ob.kglnaobj obj_name,
  lk.kgllkcnt lck_cnt,
  lk.kgllkmod lock_mode,
  lk.kgllkreq lock_req,
  s.state, s.event, s.wait_time, s.seconds_in_wait
from
  x$kgllk lk, x$kglob ob, x$ksuse ses, v$session s
where lk.kgllkhd1 in
(select kgllkhd1 from x$kgllk where kgllkreq > 0)
and ob.kglhdadr = lk.kgllkhd1
and lk.kgllkuse = ses.addr
and s.sid = ses.indx;
```

libcache1.sql

## Demo

- Create procedure – cr\_testproc.sql
- Session 1
  - exec testproc (300)
- Session 2
  - alter procedure testproc compile;
- Session 3 (SYS Session):  
select sid, state, blocking\_session, seconds\_in\_wait,  
event, p1, p1text, p1raw from v\$session where username =  
'SCOTT'

wait1.sql

## Decoding Library Cache

- x\$kgllk – Locks
  - kgllkhdl – the lock handle (address)
  - Kgllkcnt – the number of locks
  - Kgllkmod – mode of the lock
  - Kgllkreq – the requested mode on that lock
- x\$kglob ob – Objects
  - kglnaown - owner
  - Kglnaobj – name
  - Kglhdadr – the latch address
- x\$ksuse – Sessions
  - Indx – the session SID

## Mutex

- Latches contain much more information sometimes not needed
- Mutex = Mutual Exclusion
- Mutextes
  - are smaller than latches, 28 bytes instead of 110 bytes
  - take less number of instruction: ~30 instead of ~150

## 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](http://arup.blogspot.com)

Tweeter: [@ArupNanda](https://twitter.com/ArupNanda)

Facebook: [Facebook.com/ArupKNanda](https://www.facebook.com/ArupKNanda)