Rich Niemiec, TUSC
(Thanks: Scott Martin, Tirth, Andy, Nitin, John, Kevin)
Audience Knowledge

- Oracle8i Experience?
- Oracle9i Experience?
- Oracle9i RAC Experience?
- Oracle10g Experience?

Goals
- Overview of Block-level tools
- Cool things you can do

Non-Goals
- Covering ALL exceptions
Overview

• Terminology
• What you’re Waiting on
• Block Dumps & Block Information
• Transactions moving through Oracle
  – Buffer Cache
  – ITL & UNDO tie
  – Delayed block cleanout
Terminology
# Statspack - Top Wait Events

**Things to look for...**

<table>
<thead>
<tr>
<th>Wait Problem</th>
<th>Potential Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Logical Reads</td>
<td>All reads cached in memory. Includes both consistent gets and also the db block gets.</td>
</tr>
<tr>
<td>Consistent Gets</td>
<td>These are the reads of a block that are in the cache. They are NOT to be confused with consistent read (cr) version of a block in the buffer cache (usually the current version is read).</td>
</tr>
<tr>
<td>Db block gets</td>
<td>These are block gotten to be changed. MUST be the CURRENT block and not a cr block.</td>
</tr>
<tr>
<td>Db block changes</td>
<td>These are the db block gets (above) that were actually changed.</td>
</tr>
<tr>
<td>Physical Reads</td>
<td>Blocks not read from the cache. Either from disk, disk cache or O/S cache; there are also physical reads direct which bypass cache using Parallel Query (not in hit ratios).</td>
</tr>
</tbody>
</table>
## Statspack – Instance Activity

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>branch node splits</td>
<td>7,162</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>consistent gets</td>
<td>12,931,850,777</td>
<td>152,858.8</td>
<td>3,969.5</td>
</tr>
<tr>
<td>current blocks converted for CR</td>
<td>75,709</td>
<td>0.9</td>
<td>0.0</td>
</tr>
<tr>
<td>db block changes</td>
<td>343,632,442</td>
<td>4,061.9</td>
<td>105.5</td>
</tr>
<tr>
<td>db block gets</td>
<td>390,323,754</td>
<td>4,613.8</td>
<td>119.8</td>
</tr>
<tr>
<td>hot buffers moved to head of LRU</td>
<td>197,262,394</td>
<td>2,331.7</td>
<td>60.6</td>
</tr>
<tr>
<td>leaf node 90-10 splits</td>
<td>26,429</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>leaf node splits</td>
<td>840,436</td>
<td>9.9</td>
<td>0.3</td>
</tr>
<tr>
<td>logons cumulative</td>
<td>21,369</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>physical reads</td>
<td>504,643,275</td>
<td>5,965.1</td>
<td>154.9</td>
</tr>
<tr>
<td>physical writes</td>
<td>49,724,268</td>
<td>587.8</td>
<td>15.3</td>
</tr>
<tr>
<td>session logical reads</td>
<td>13,322,170,917</td>
<td>157,472.5</td>
<td>4,089.4</td>
</tr>
<tr>
<td>sorts (disk)</td>
<td>4,132</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>sorts (memory)</td>
<td>7,938,085</td>
<td>93.8</td>
<td>2.4</td>
</tr>
<tr>
<td>sorts (rows)</td>
<td>906,207,041</td>
<td>10,711.7</td>
<td>278.2</td>
</tr>
<tr>
<td>table fetch continued row</td>
<td>25,506,365</td>
<td>301.5</td>
<td>7.8</td>
</tr>
<tr>
<td>table scans (long tables)</td>
<td>111</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>table scans (short tables)</td>
<td>1,543,085</td>
<td>18.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Tuning the RAC Cluster Interconnect
RAC issues are the same times TWO!

Top 5 Timed Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Waits</th>
<th>Time (s)</th>
<th>Ela Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>global cache cr request</td>
<td>820</td>
<td>154</td>
<td>72.50</td>
</tr>
<tr>
<td>CPU time</td>
<td></td>
<td>54</td>
<td>25.34</td>
</tr>
<tr>
<td>global cache null to x</td>
<td>478</td>
<td>1</td>
<td>.52</td>
</tr>
<tr>
<td>control file sequential read</td>
<td>600</td>
<td>1</td>
<td>.52</td>
</tr>
<tr>
<td>control file parallel write</td>
<td>141</td>
<td>1</td>
<td>.28</td>
</tr>
</tbody>
</table>

• Transfer times excessive from other instances in the cluster to this instance.

• Could be due to network problems or buffer transfer issues.
Current & CR Versions

- Buffer hash table x$bh holds headers (hash chain protected by a CBC latch) point to db_block buffers in memory.
- For a given block - Only one block is CURRENT and no more than 5 other CR versions of the block (as of V9).
- For DML, you need the CURRENT version.
- For query, you can use the CURRENT version if not being used and/or build a CONSISTENT READ (CR) version by applying and undo needed. This may include reading the ITL, mapping to the UNDO HEADER, but the ITL also maps directly to the UNDO BLOCK and applying the UNDO to get the correct CR version that you need.
- Links for the LRU & LRU-W (working set used for buffer replacement) are maintained in the buffer headers.
Biggest Problems

- The SQL, of course… especially reads of full indexes tables and others.
- Hot blocks… hot blocks can cause latching issues. Bad SQL or bad indexes causes hot blocks (scanning through the same large index). **Improved in 10g (shared latches).**
- Not enough freelists or not using ASSM.
- Not enough initrans for multiple DML to the same block (pctfree not high enough to auto-generate). Or too many (each ITL costs 24 bytes).
- Slow I/O subsystem or poor disk caching or not enough paths and readers/writers colliding.
- Not on latest version so can’t use great new features!
What are you Waiting on?
<table>
<thead>
<tr>
<th>Wait Problem</th>
<th>Potential Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Read</td>
<td>Indicates many index reads – tune the code (especially joins); Faster I/O</td>
</tr>
<tr>
<td>Scattered Read</td>
<td>Indicates many full table scans – tune the code; cache small tables; Faster I/O</td>
</tr>
<tr>
<td>Free Buffer</td>
<td>Increase the DB_CACHE_SIZE; shorten the checkpoint; tune the code to get less dirty blocks, faster I/O, use multiple DBWR’s.</td>
</tr>
<tr>
<td>Buffer Busy</td>
<td>Segment Header – Add freelists (if inserts) or freelist groups (esp. RAC). Use ASSM.</td>
</tr>
<tr>
<td>Wait Problem</td>
<td>Potential Fix</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Buffer Busy</td>
<td>Data Block – Separate ‘hot’ data; potentially use reverse key indexes; fix queries to reduce the blocks popularity, use smaller blocks, I/O, Increase initrans and/or maxtrans (this one’s debatable) Reduce records per block.</td>
</tr>
<tr>
<td>Buffer Busy</td>
<td>Undo Header – Add rollback segments or increase size of segment area (auto undo)</td>
</tr>
<tr>
<td>Buffer Busy</td>
<td>Undo block – Commit more (not too much) Larger rollback segments/area. Try to fix the SQL.</td>
</tr>
</tbody>
</table>
### Statspack - Top Wait Events

**Things to look for...**

<table>
<thead>
<tr>
<th>Wait Problem</th>
<th>Potential Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enqueue - ST</td>
<td>Use LMT’s or pre-allocate large extents</td>
</tr>
<tr>
<td>Enqueue - HW</td>
<td>Pre-allocate extents above HW (high water mark.)</td>
</tr>
<tr>
<td>Enqueue – TX</td>
<td>Increase initrans and/or maxtrans (TX4) on (transaction) the table or index. Fix locking issues if TX6. Bitmap (TX4) &amp; Duplicates in Index (TX4).</td>
</tr>
<tr>
<td>Enqueue - TM (trans. mgmt.)</td>
<td>Index foreign keys; Check application locking of tables. DML Locks.</td>
</tr>
<tr>
<td>Wait Problem</td>
<td>Potential Fix</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CBC Latches</td>
<td>Cache Buffers Chains Latches – Reduce the length of the hash chain (less copies) by reducing block’s popularity. Increase the latches by increasing buffers. Use Oracle SQ generator.</td>
</tr>
<tr>
<td>LRU Chain Latch</td>
<td>This latch protects the LRU list when a user needs the latch to scan the LRU chain for a buffer. When a dirty buffer is encountered it is linked to the LRU-W. When adding, moving, or removing a buffer this latch is needed.</td>
</tr>
</tbody>
</table>
# Block Dumps

<table>
<thead>
<tr>
<th>Dec Hx Oct Htm Chr</th>
<th>Dec Hx Oct Htm Chr</th>
<th>Dec Hx Oct Htm Chr</th>
<th>Dec Hx Oct Htm Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 000 NULL</td>
<td>0 0 000 NULL</td>
<td>0 0 000 NULL</td>
<td>0 0 000 NULL</td>
</tr>
<tr>
<td>1 1 001 SOH</td>
<td>1 1 001 SOH</td>
<td>1 1 001 SOH</td>
<td>1 1 001 SOH</td>
</tr>
<tr>
<td>2 2 002 STX</td>
<td>2 2 002 STX</td>
<td>2 2 002 STX</td>
<td>2 2 002 STX</td>
</tr>
<tr>
<td>3 3 003 ETX</td>
<td>3 3 003 ETX</td>
<td>3 3 003 ETX</td>
<td>3 3 003 ETX</td>
</tr>
<tr>
<td>4 4 004 EOT</td>
<td>4 4 004 EOT</td>
<td>4 4 004 EOT</td>
<td>4 4 004 EOT</td>
</tr>
<tr>
<td>5 5 005 ENQ</td>
<td>5 5 005 ENQ</td>
<td>5 5 005 ENQ</td>
<td>5 5 005 ENQ</td>
</tr>
<tr>
<td>6 6 006 ACK</td>
<td>6 6 006 ACK</td>
<td>6 6 006 ACK</td>
<td>6 6 006 ACK</td>
</tr>
<tr>
<td>7 7 007 BEL</td>
<td>7 7 007 BEL</td>
<td>7 7 007 BEL</td>
<td>7 7 007 BEL</td>
</tr>
<tr>
<td>8 8 010 BS</td>
<td>8 8 010 BS</td>
<td>8 8 010 BS</td>
<td>8 8 010 BS</td>
</tr>
<tr>
<td>9 9 011 HT</td>
<td>9 9 011 HT</td>
<td>9 9 011 HT</td>
<td>9 9 011 HT</td>
</tr>
<tr>
<td>A A 012 LF</td>
<td>A A 012 LF</td>
<td>A A 012 LF</td>
<td>A A 012 LF</td>
</tr>
<tr>
<td>B B 013 VT</td>
<td>B B 013 VT</td>
<td>B B 013 VT</td>
<td>B B 013 VT</td>
</tr>
<tr>
<td>C C 014 FF</td>
<td>C C 014 FF</td>
<td>C C 014 FF</td>
<td>C C 014 FF</td>
</tr>
<tr>
<td>D D 015 CR</td>
<td>D D 015 CR</td>
<td>D D 015 CR</td>
<td>D D 015 CR</td>
</tr>
<tr>
<td>E E 016 SO</td>
<td>E E 016 SO</td>
<td>E E 016 SO</td>
<td>E E 016 SO</td>
</tr>
<tr>
<td>F F 017 ST</td>
<td>F F 017 ST</td>
<td>F F 017 ST</td>
<td>F F 017 ST</td>
</tr>
<tr>
<td>10 020 DLE</td>
<td>10 020 DLE</td>
<td>10 020 DLE</td>
<td>10 020 DLE</td>
</tr>
<tr>
<td>17 021 DC1</td>
<td>17 021 DC1</td>
<td>17 021 DC1</td>
<td>17 021 DC1</td>
</tr>
<tr>
<td>18 022 DC2</td>
<td>18 022 DC2</td>
<td>18 022 DC2</td>
<td>18 022 DC2</td>
</tr>
<tr>
<td>19 023 DC3</td>
<td>19 023 DC3</td>
<td>19 023 DC3</td>
<td>19 023 DC3</td>
</tr>
<tr>
<td>20 024 DC4</td>
<td>20 024 DC4</td>
<td>20 024 DC4</td>
<td>20 024 DC4</td>
</tr>
<tr>
<td>21 025 NAK</td>
<td>21 025 NAK</td>
<td>21 025 NAK</td>
<td>21 025 NAK</td>
</tr>
<tr>
<td>22 026 SYN</td>
<td>22 026 SYN</td>
<td>22 026 SYN</td>
<td>22 026 SYN</td>
</tr>
<tr>
<td>23 027 ETB</td>
<td>23 027 ETB</td>
<td>23 027 ETB</td>
<td>23 027 ETB</td>
</tr>
<tr>
<td>24 030 CAN</td>
<td>24 030 CAN</td>
<td>24 030 CAN</td>
<td>24 030 CAN</td>
</tr>
<tr>
<td>25 031 EM</td>
<td>25 031 EM</td>
<td>25 031 EM</td>
<td>25 031 EM</td>
</tr>
<tr>
<td>26 032 SUB</td>
<td>26 032 SUB</td>
<td>26 032 SUB</td>
<td>26 032 SUB</td>
</tr>
<tr>
<td>27 033 ESC</td>
<td>27 033 ESC</td>
<td>27 033 ESC</td>
<td>27 033 ESC</td>
</tr>
<tr>
<td>28 034 FS</td>
<td>28 034 FS</td>
<td>28 034 FS</td>
<td>28 034 FS</td>
</tr>
<tr>
<td>29 035 GS</td>
<td>29 035 GS</td>
<td>29 035 GS</td>
<td>29 035 GS</td>
</tr>
<tr>
<td>30 036 RS</td>
<td>30 036 RS</td>
<td>30 036 RS</td>
<td>30 036 RS</td>
</tr>
<tr>
<td>31 037 US</td>
<td>31 037 US</td>
<td>31 037 US</td>
<td>31 037 US</td>
</tr>
</tbody>
</table>

Source: [LeslapTables.com](http://LeslapTables.com)
Last Resort - Block Dumps

```sql
SQL> desc emp1
Name | Null? | Type
---------------------------------------- -------- ----------------------------
EMPNO |       | NUMBER(4)
ENAME |       | VARCHAR2(10)
JOB   |       | VARCHAR2(9)
MGR   |       | NUMBER(4)
HIREDATE |   | DATE
SAL   |       | NUMBER(7,2)
COMM  |       | NUMBER(7,2)
DEPTNO |     | NUMBER(2)
```
Last Resort - Block Dumps

```sql
select *
from emp
where ename = 'MILLER';
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>JOB</th>
<th>MGR</th>
<th>HIREDATE</th>
<th>SAL</th>
<th>COMM</th>
<th>DEPTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7934</td>
<td>MILLER</td>
<td>CLERK</td>
<td>7782</td>
<td>23-JAN-82</td>
<td>1300</td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>
Last Resort - Block Dumps

```
select file_id, block_id, blocks
from dba_extents
where segment_name = 'EMP'
and owner = 'SCOTT';
```

<table>
<thead>
<tr>
<th>FILE_ID</th>
<th>BLOCK_ID</th>
<th>BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50465</td>
<td>3</td>
</tr>
</tbody>
</table>
ALTER SYSTEM DUMP DATAFILE 5 BLOCK 50465
/
ALTER SYSTEM DUMP DATAFILE 5 BLOCK 50466
/
ALTER SYSTEM DUMP DATAFILE 5 BLOCK 50467
/
Or…

ALTER SYSTEM DUMP DATAFILE 5 BLOCK MIN 50465 BLOCK MAX 50467;

(Puts output in user_dump_dest)
Block Dump... Getting the block number

```sql
select rowid, empno,
    dbms_rowid.rowid_relative_fno(rowid) fileno,
    dbms_rowid.rowid_block_number(rowid) blockno,
    dbms_rowid.rowid_row_number(rowid) rowno, rownum,
    rpad(to_char(dbms_rowid.rowid_block_number(rowid), 'FM0xxxxxxx') || '.' ||
          to_char(dbms_rowid.rowid_row_number (rowid), 'FM0xxx' ) || '.' ||
          to_char(dbms_rowid.rowid_relative_fno(rowid), 'FM0xxx' ), 18) myrid
from emp1;
```

<table>
<thead>
<tr>
<th>ROWID</th>
<th>EMPNO</th>
<th>FILENO</th>
<th>BLOCKNO</th>
<th>ROWNO</th>
<th>ROWNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYRID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAMfcAABAAAN0KAAA</td>
<td>7369</td>
<td>1</td>
<td>56586</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0000dd0a.0000.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAMfcAABAAAN0KAAB</td>
<td>7499</td>
<td>1</td>
<td>56586</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0000dd0a.0001.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAMfcAABAAAN0KAAC</td>
<td>7521</td>
<td>1</td>
<td>56586</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0000dd0a.0002.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Block Dumps: Top Section
Block Dumps – Top Section

*** 2005-04-08 23:18:49.226
Start dump data blocks tsn: 0 file#: 1 minblk 56650 maxblk 56650
buffer tsn: 0 rdba: 0x0040dd4a (1/56650)
scn: 0x0000.003dfa58 seq: 0x01 flg: 0x00 tail: 0xfa580601
frmt: 0x02 chkval: 0x0000 type: 0x06=trans data
Block header dump: 0x0040dd4a
Object id on Block? Y
seg/obj: 0xce1c csc: 0x00.3dfa58 itc: 2 flg: O typ: 1 - DATA
fsl: 0 fnx: 0x0 ver: 0x01

Scn block was last changed at
ACTUAL Database Block Address

Changes to block w/i scn

Blocks that were dumped

#ITL’s (see next slide)
## Block Dumps – Top Section

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x0004.010.00000fba</td>
<td>0x0080003d.08b5.10</td>
<td>----</td>
<td>4 fsc 0x009d.00000000</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.016.00000fae</td>
<td>0x008000cc.08af.34</td>
<td>C---</td>
<td>0 scn 0x0000.003deb5b</td>
</tr>
</tbody>
</table>

### ITL – 2 Interested Transaction Lists

- **Rows Locked:** 4 rows deleted for this xid in this block.

### UBA:

- **File.block(Undo dba).sequence.record**
- **Undo block address where last change is recorded.**
## Block Dumps – Top Section

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x0004.010.00000fba</td>
<td>0x0080003d.08b5.10</td>
<td>----</td>
<td>4 fsc 0x009d.00000000</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.016.00000fae</td>
<td>0x008000cc.08af.34</td>
<td>C---</td>
<td>0 scn 0x0000.003deb5b</td>
</tr>
</tbody>
</table>

**Flag:** No flag set then it’s uncommitted (----)/(CBUT)

**C---** = Committed

**-B--** = The UBA contains undo for this itl

**--U-** = Committed (scn is upper bound)…used by fast commits & delayed block cleanout has not occurred

**---T** = Transaction active at block cleanout SCN

**-C-U-** = Block cleaned by delayed block cleanout, and the rollback segment info overwritten. The scn will show the lowest scn that could be regenerated by the rollback segment.
### Block Dumps – Top Section

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x0004.010.00000fba</td>
<td>0x0080003d.08b5.10</td>
<td>----</td>
<td>4 fsc 0x009d.00000000</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.016.00000fae</td>
<td>0x008000cc.08af.34</td>
<td>C---</td>
<td>0 scn 0x0000.003deb5b</td>
</tr>
</tbody>
</table>

**Scn/fsc:**
- **Fsc** = free space Credit = 9d (hex) = 157 (decimal) bytes
- **Scn** = System change (commit) number
Block Dumps – Top Section

data_block_dump, data header at 0x5a1125c

================

tsiz: 0x1fa0  Total Size = fa0
1 (hex) = 1 (decimal) x 16x16x16 = 4096
F (hex) = 15 (decimal) x 16x16 = 3840
A (hex) = 10 (decimal) x 16 = 160
0 (hex) = 0 (decimal) x 1 = 0
4096 + 3840 + 160 + 0 = 8096 (about 8K)

hsiz: 0x2e  Header Size = 2e = 46 bytes

pbl: 0x5a1125c  Pointer to block buffer holding the block

bdba: 0x0040dd4a  (Relative) database block address
ntab=1
nrow=14
frre=-1
fsbo=0x2e
fseo=0x18fb
avsp=0x1d3b
tosp=0x1de0

0xe:pti[0]  nrow=14  offs=0  Table Info
0x12:pri[0]  offs=0x18fb  (6395) Row Info Record 0
0x14:pri[1]  offs=0x1921  (6433) Row Info Record 1
...

block_row_dump:  Row Data is Next!

Number of tables= 1 (2+ for clusters)
Number of rows = 14
First free row index entry – 1 (add 1)
Free space begin offset
Free space end offset
Available block space = 1d3b = 7483
Space avail. post commit=1de0=7648
The familiar emp table Terlingua software
* Terlingua Software
Row directory

* Terlingua Software
Interested Transaction List (ITL)

* Terlingua Software
Block Dump: Data Section
Block Dumps – output from udump

<table>
<thead>
<tr>
<th>col 0</th>
<th>col 1</th>
<th>col 2</th>
<th>col 3</th>
<th>col 4</th>
<th>col 5</th>
<th>col 6</th>
<th>col 7</th>
</tr>
</thead>
</table>
Block Dumps – Data Section

**DUMP OUTPUT:**

- **tab 0, row 13, @0x1b0b**
- **tl: 39 fb: --H-FL-- lb: 0x0 cc: 8** (row header)

**Table = this data is for table 0**
**Row 13 = 14th Row (0-13 total rows)**
**Offset: 1b0b (in Hex) – Offset from header**
**tl: Total bytes of row plus the header = 39**
DUMP OUTPUT:

tab 0, row 13, @0x1b0b

tl: 39  fb: --H-FL--  lb: 0x0  cc: 8

fb: --H-FL-- = flag byte; ( -KCHDFLPN)

H = Head of row piece, F = First data piece, L=Last piece
D = Deleted; P= First column continues from previous piece
   (chaining) ; N= Last column continues in next piece;
K = Cluster Key; C = Cluster table member
Rows point back to itl

2c (Hex) = 00101100 (Binary) = --H-FL-- (flags)
DUMP OUTPUT:

tab 0, row 13, @0x1b0b

tl: 39 fb: --H-FL-- lb: 0x0 cc: 8

Lb: lock byte is 1+ if this row is locked = 0 (unlocked)
cc: Column count = 8
DUMP OUTPUT (Deleted Row):

block_row_dump:

tab 0, row 0, @0x18fb
 tl: 2 fb: --HDFL-- lb: 0x1

tab 0, row 1, @0x1921
 tl: 2 fb: --HDFL-- lb: 0x1

Rows 1 & 2 have been deleted! No row data is visible for the columns.
Block Dumps – Data Section

DUMP OUTPUT - EMPNO:
col 0: [ 3] c2 50 23

Hex to Decimal: Col0 = EMPNO = 7934
50 (Hex) = 80 (Decimal) – 1 = 79
23 (Hex) = 35 (Decimal) – 1 = 34
c2: Number in the thousands (c2 is exponent)
**Block Dumps – Data Section**

**DUMP OUTPUT - ENAME:**

**col 1: [ 6] 4d 49 4c 4c 45 52**

Hex to Character:  
Col1 = ENAME = MILLER

4d (Hex) = M (Character)
49 (Hex) = I (Character)
4c (Hex) = L (Character)
4c (Hex) = L (Character)
45 (Hex) = E (Character)
52 (Hex) = R (Character)
Block Dumps – Data Section

DUMP OUTPUT - JOB:
col 2: [ 5] 43 4c 45 52 4b

Hex to Character:  
Col2 = JOB = CLERK

43 (Hex) = C (Character)
4c (Hex) = L (Character)
45 (Hex) = E (Character)
52 (Hex) = R (Character)
4b (Hex) = K (Character)
DUMP OUTPUT, MGR:
col 3: [3] c2 4e 53

Hex to Decimal:
Col3 = MGR = 7782
4e (Hex) = 78 (Decimal) – 1 = 77
53 (Hex) = 83 (Decimal) – 1 = 82
Block Dumps – Data Section

DUMP OUTPUT - HIREDATE:

col 4: [ 7] 77 b6 01 17 01 01 01

Hex to Decimal: Col4 = HIREDATE = 23-JAN-82

77 (Hex) =  119 (Decimal) – 100 = 19
B6 (Hex) =  182 (Decimal) – 100 = 82
01(Hex) =  1 (Decimal) <month>
17 (Hex) =  23 (Decimal)
01 01 01 (Hex) = This is the Hour, Minute, Second
(none were entered when the date was entered…default)
DUMP OUTPUT - SAL:

col 5: [2] c2 0e

Hex to Decimal: Col5 = SAL = 1300

0e (Hex) = 14 (Decimal) - 1 = 13

c2 = add two zero’s
Block Dumps – Data Section

DUMP OUTPUT - COMM:

col 6: *NULL*

Hex to Decimal: Col6 = COMM = NULL

NULL = NULL
**DUMP OUTPUT - DEPTNO:**

col 7: [2] c1 0b

Hex to Decimal:  
Col7 = DEPTNO = 10

0B (Hex) = 11 (Decimal) - 1 = 10

c1 = number in the tens
Block Dump:
Data Section – Other Ways
Block Dump using SELECT dump()

```sql
select dump(ename) from emp1
where ename='MILLER';
```

**DUMP(ENAME)**

```
- Typ=1 Len=6: 77,73,76,76,69,82

Types: 1=varchar; 2=number; 12=date; 23=raw
```
Block Dump using SELECT dump()

Typ=1 Len=6: 77,73,76,76,69,82

Decimal to Character: ENAME = MILLER

77 (Decimal) = M (Character)
73 (Decimal) = I (Character)
76 (Decimal) = L (Character)
76 (Decimal) = L (Character)
69 (Decimal) = E (Character)
82 (Decimal) = R (Character)
Block Dump using SELECT dump()
(convert it to HEX if you want)

```sql
select dump(ename.16) from emp1
where ename='MILLER';
```

**DUMP(ENAME,16)**

```
Typ=1 Len=6: 4d,49,4c,4c,45,52
```

Types: 1=varchar; 2=number; 12=date; 23=raw
Block Dump using SELECT dump()
(Can even get the ename from the HEX!)

```sql
select dump(ename,16), ename
from emp1
where dump(ename,16) like '%4d,49,4c,4c,45,52';
```

<table>
<thead>
<tr>
<th>DUMP(ENAME,16)</th>
<th>ENAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typ=1 Len=6: 4d,49,4c,4c,45,52</td>
<td>MILLER</td>
</tr>
</tbody>
</table>
Block Dump using SELECT dump()

```
select dump(empno)
from emp1
where ename='MILLER';
```

**DUMP(EMPNO)**

```
Typ=2 Len=3: 194,80,35
```

Decimal to number:  

EMPNO = 7934

194 (Decimal) = c2 (Hex)

80 (Decimal) – 1 = 79

35 (Decimal) – 1 = 34
Block Dump using SELECT dump()

```sql
select dump(hiredate)
from emp1
where ename='MILLER';
```

**DUMP(HIREDATE)**

```
Typ=12 Len=7: 119,182,1,23,1,1,1
```

**Decimal to Date:**  
HIREDATE = 23-JAN-82

119 (decimal) – 100 = 19 (century)
182 (decimal) – 100 = 82 (year)
1 = Month (January) ; 23 = Day
Transactions
Moving through Oracle
Working with Hash Buckets And Buffer Headers (not buffers)

• Users asks for a specific data block address.
• This is hashed with a hashing algorithm and placed in the hash bucket that it hashes to.
• It walks the hash chain using the cache buffers chain latch to find the block that it needs (curr or cr).
• There can be many versions of each block
_DB_BLOCK_HASH_BUCKETS
and hashing data block addresses

Example: _DB_BLOCK_HASH_BUCKETS
(shouldn’t have to change this in Oracle9i or 10g)

- Buffer hash table (x$bh) has all buffer headers for all db_block buffers.
- Buffer header ties to memory base address of the buffer.
- Buckets usually set to Prime(2*db_block_buffers)
- A prime number is often used to avoid hashing anomalies
- Objects dba (class) is hashed to a hash bucket on the hash chain
- Get enough hash buckets (_db_block_hash_buckets)
- Blocks assigned to a hash bucket and onto the hash chain
- Could have multiple blocks hashed to same chain (if both hot-issues)
- Can have multiple versions of a block on same chain
- When block is replaced (based on LRU chain) new block comes in and could be (probably will be) hashed to a different hash chain.
**Example: Emp Table**

Consider a user querying `emp`:

- First block of `emp` may go to chain #1
- Second block of `emp` may go to chain #55
- If second block of `emp` is updated and also has several readers than we’ll get more copies. **LRBA – Lowest Redo Block Address** (last redo applied) for dirty block.
- Chain #55 may now have a current block and 2 CR blocks all with the same dba (data block address)
- For a given block - Only one block is CURRENT and no more than 5 other CR versions of the block (as of V9).
- All buffer headers tie to LRU, LRU-W and other LRU’s (many in 10g) used for buffer replacement.
Additional LRU’s / Faster!!

- **LRU**  Main block replacement list
- **LRU-W**  Old dirty buffers and reco/temp
- **LRU-P**  Ping Buffer list / RAC
- **LRU-XO**  Buffers to be written for drop/truncate
- **LRU-XR**  Buffers to be written for reuse range
- **Thread CKPT**  Thread Checkpoint Queue
- **File CKPT**  File Checkpoint Queue
- **Reco CKPT**  Reco Checkpoint
- **LRU-MAIN & LRU-AUX** help LRU
**Query all buffer headers (state):**

```sql
col status for a6
select state,
    decode(state, 0, 'FREE', /* not currently in use */
        1, 'XCUR', /* held exclusive by this instance */
        2, 'SCUR', /* held shared by this instance */
        3, 'CR', /* only valid for consistent read */
        4, 'READ', /* is being read from disk */
        5, 'MREC', /* in media recovery mode */
        6, 'IREC', /* in instance(crash) recovery mode */
        7, 'WRITE', /* being written */
        8, 'PIN') status, count(*) /* pinned */
from x$bh
group by state;
```

<table>
<thead>
<tr>
<th>STATE</th>
<th>STATUS</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XCUR</td>
<td>2001</td>
</tr>
<tr>
<td>3</td>
<td>CR</td>
<td>3</td>
</tr>
</tbody>
</table>
EMP1 is Block#: 56650
(all rows are in this block)

```
select rowid, empno,
    dbms_rowid.rowid_relative_fno(rowid) fileno,
    dbms_rowid.rowid_block_number(rowid) blockno,
    dbms_rowid.rowid_row_number(rowid) rowno, rownum,
    rpad(to_char(dbms_rowid.rowid_block_number(rowid), 'FM0xxxxxxx') || '.' ||
         to_char(dbms_rowid.rowid_row_number(rowid), 'FM0xxx') || '.' ||
         to_char(dbms_rowid.rowid_relative_fno(rowid), 'FM0xxx'), 18) myrid
from emp1;
```

<table>
<thead>
<tr>
<th>ROWID</th>
<th>EMPNO</th>
<th>FILENO</th>
<th>BLOCKNO</th>
<th>ROWNO</th>
<th>ROWNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAM4cAABAAAN1KAAA</td>
<td>7369</td>
<td>1</td>
<td>56650</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0000dd4a.0000.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAM4cAABAAAN1KAAB</td>
<td>7499</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0000dd4a.0001.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAM4cAABAAAN1KAAN</td>
<td>7934</td>
<td>1</td>
<td>56650</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>0000dd4a.000d.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14 rows selected.
Let’s watch the EMP1 buffer header
(So far it’s clean and only 1 copy)

```
select  lrba_seq, state, dbarfil, dbablk, tch, flag, hscn_bas, cr_scn_bas,
       decode(bitand(flag,1), 0, 'N', 'Y') dirty,  /* Dirty bit */
       decode(bitand(flag,16), 0, 'N', 'Y') temp,  /* temporary bit */
       decode(bitand(flag,1536), 0, 'N', 'Y') ping, /* ping (to shared or null) bit */
       decode(bitand(flag,16384), 0, 'N', 'Y') stale, /* stale bit */
       decode(bitand(flag,65536), 0, 'N', 'Y') direct, /* direct access bit */
       decode(bitand(flag,1048576), 0, 'N', 'Y') new /* new bit */
from    x$bh
where   dbablk = 56650
order by dbablk;
```

```
<table>
<thead>
<tr>
<th>LRBA_SEQ</th>
<th>STATE</th>
<th>DBARFIL</th>
<th>DBABLK</th>
<th>TCH</th>
<th>FLAG</th>
<th>HSCN_BAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR_SCN_BAS</td>
<td>D</td>
<td>T</td>
<td>P</td>
<td>S</td>
<td>D</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
```

63
Only ONE block on the Hash Chain!
Let’s watch the EMP1 buffer header (Delete a row)

delete from emp1
where comm = 0;

one row deleted.
Let's watch the EMP1 buffer header
(Make some changes 2 copies)

```
select  lrba_seq, state, dbarfil, dbablk, tch, flag, hscn_bas, cr_scn_bas,
        decode(bitand(flag,1), 0, 'N', 'Y') dirty, /* Dirty bit */
        decode(bitand(flag,16), 0, 'N', 'Y') temp,  /* temporary bit */
        decode(bitand(flag,1536), 0, 'N', 'Y') ping, /* ping (to shared or null) bit */
        decode(bitand(flag,16384), 0, 'N', 'Y') stale, /* stale bit */
        decode(bitand(flag,65536), 0, 'N', 'Y') direct, /* direct access bit */
        decode(bitand(flag,1048576), 0, 'N', 'Y') new  /* new bit */
from x$bh
where dbablk = 56650
order by dbablk;
```

<table>
<thead>
<tr>
<th>LRBA_SEQ</th>
<th>STATE</th>
<th>DBARFIL</th>
<th>DBABLK</th>
<th>TCH</th>
<th>FLAG</th>
<th>HSCN_BAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_SCN_BAS</td>
<td>D T P S D N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>8200</td>
<td>4294967295</td>
</tr>
<tr>
<td>0 N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>2</td>
<td>524288</td>
<td>0</td>
</tr>
<tr>
<td>4347881 N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hash Chain is now TWO! One is a CR and the other is Current.
V$Transaction now has our record
(created when transactions have undo)

SELECT t.addr, t.xidusn USN, t.xidslot SLOT, t.xidsqn SQL, t.status,
       t.used_ublk UBLK, t.used_urec UREC, t.log_io LOG,
       t.phy_io PHY, t.cr_get, t.cr_change CR CHA
FROM v$transaction t, v$session s
WHERE t.addr = s.taddr;

<table>
<thead>
<tr>
<th>ADDR</th>
<th>USN</th>
<th>SLOT</th>
<th>SQL</th>
<th>STATUS</th>
<th>UBLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>69E50E5C</td>
<td>5</td>
<td>42</td>
<td>652</td>
<td>ACTIVE</td>
<td>1</td>
</tr>
</tbody>
</table>

USN is the Undo Segment Number (rollback segment ID)
SLOT is the slot number in the rollback segment’s transaction table.
SQN (Wrap) is the sequence number for the transaction.
USN+SLOT+SQN are the three values that uniquely identifies a transaction XID.
V$Transaction

UBAFIL is the file for last undo entry
UBLK is block for last undo entry (find out how many undo blocks).
UBASQN is the sequence no of the last entry.
UREC is the record number of the block (shows how many table and index entries the transaction has inserted, updated or deleted.

If you are doing an INSERT or DELETE, then you will see that UREC is set to <number of indexes for this table> + how many rows you inserts/deletes. If you UPDATE a column then UREC will be set to <number of indexes that his column belongs to> * 2 + number of updated rows (so if the column belongs to no index, then UREC is set to the number of rows that was updated).

If USED_UBLK and USED_UREC are decreasing each time you query, then the transaction is rolling back. When USED_UREC zero, the rollback is finished.
### Dump the block

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x0005.02a.0000028c</td>
<td>0x008000af:02b6.01</td>
<td>----</td>
<td>1</td>
<td>fsc 0x0029.00000000</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.016.00000fae</td>
<td>0x008000cc.08af.34</td>
<td>C---</td>
<td>0</td>
<td>scn 0x0000.003deb5b</td>
</tr>
</tbody>
</table>

**ITL – 2 Interested Transaction Lists**

**Transaction ID**
- Undo 5 = 5 (decimal)
- Slot 2a = 42 (decimal)
- SEQ 28C = 652

**Committed Transaction**

**UBA:**
- File.block.sequence.record
- Undo block address where last change is recorded.

**The row I deleted is still locked; fsc is 0x29 = 41 bytes**
### Table Content

<table>
<thead>
<tr>
<th>LRBA_SEQ</th>
<th>STATE</th>
<th>DBARFTL</th>
<th>DBABLK</th>
<th>TCH</th>
<th>FLAG</th>
<th>HSCN_BAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_SCN_BAS</td>
<td>D</td>
<td>T</td>
<td>P</td>
<td>S</td>
<td>D</td>
<td>N</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350120</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350105</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>365</td>
<td>1</td>
<td>1</td>
<td>56650</td>
<td>7</td>
<td>33562633</td>
<td>4350121</td>
</tr>
<tr>
<td>0</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4350103</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350089</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350087</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
Hash Chain is now SIX long!
Five CR and the one Current.
Why only 6 versions of a Block?

```
select a.ksppinm, b.ksppstvl, b.ksppstdf, a.ksppdesc
from x$ksppi a, x$ksppcv b
where a.indx = b.indx
and substr(ksppinm,1,1) = '_'
and ksppinm like '%&1%'
order by ksppinm;
```

 KSPPINM
-----------------
 KSPPSTVL
-----------------
 KSPPSTDF
--------
 KSPPDESC
-----------------
 _db_block_max_cr_dba
 6
 TRUE
Maximum Allowed Number of CR buffers per dba
What happens after we roll everything back – x$bh Still an LRBA:

<table>
<thead>
<tr>
<th>LRBA_SEQ</th>
<th>STATE</th>
<th>DBARFIL</th>
<th>DBABLK</th>
<th>TCH</th>
<th>FLAG</th>
<th>HSCN_BAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_SCN_BAS</td>
<td>D T P S D N</td>
<td>----------</td>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350120</td>
<td>N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350105</td>
<td>N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>1</td>
<td>1</td>
<td>56650</td>
<td>11</td>
<td>35659777</td>
<td>4350702</td>
</tr>
<tr>
<td>0</td>
<td>Y N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350103</td>
<td>N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524416</td>
<td>0</td>
</tr>
<tr>
<td>4350089</td>
<td>N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>56650</td>
<td>1</td>
<td>524288</td>
<td>0</td>
</tr>
<tr>
<td>4350087</td>
<td>N N N N N N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 rows selected.
User 1 – Updates Row# 1&2
User 2 updates Row 3

• User1 updates a row with an insert/update/delete – an ITL is opened and xid tracks it in the data block.

• The xid ties to the UNDO header block which ties to the UNDO data block for undo.

• If user2 wants to query the row, they create a clone and rollback the transaction going to the undo header and undo block.

• If user3 wants to update same row (they wait). If user 3 wants to update different row then they open a second ITL with an xid that maps to an undo header that maps to an undo block.
Create EMP2 (‘MILLER’/’ALLEN’)

create table emp2
as select * from emp1
where ename in (‘MILLER','ALLEN’);

select empno, ename, job
from emp2;

EMPNO  ENAME  JOB
----------  --------  ----------
  7499  ALLEN    SALESMAN
  7934  MILLER  CLERK
Get the Blockno for EMP2

```sql
select rowid, empno,
    dbms_rowid.rowid_relative_fno(rowid) fileno,
    dbms_rowid.rowid_block_number(rowid) blockno,
    dbms_rowid.rowid_row_number(rowid) rowno, rownum,
    rpad(to_char(dbms_rowid.rowid_block_number(rowid), 'FM0xxxxxx') || '.' ||
    to_char(dbms_rowid.rowid_row_number(rowid), 'FM0xxx') || '.' ||
    to_char(dbms_rowid.rowid_relative_fno(rowid), 'FM0xxx'), 18) myrid
from emp2;
```

<table>
<thead>
<tr>
<th>ROWID</th>
<th>EMPNO</th>
<th>FILENO</th>
<th>BLOCKNO</th>
<th>ROWNO</th>
<th>ROWNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYRID</td>
<td>------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>AAANB2AABAAAOHSAAA</td>
<td>7499</td>
<td>1</td>
<td>57810</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0000e1d2.0000.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAANB2AABAAAOHSAAB</td>
<td>7934</td>
<td>1</td>
<td>57810</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0000e1d2.0001.0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dump the EMP2 block (Partial)

*Alter system dump datafile 2 block 57810; System Altered.*

Start dump data blocks tsn: 0 file#: 1 minblk **57810** maxblk 57810
buffer tsn: 0 rdba: 0x0040e1d2 (1/57810)
scn: 0x0000.00432370 seq: 0x02 flg: 0x04 tail: 0x23700602
fmt: 0x02 chkval: 0xb205 type: 0x06=trans data
Block header dump: 0x0040e1d2
Object id on Block? Y
seg/obj: 0xd076 csc: 0x00.43236f itc: 3 flg: - typ: 1 - DATA
fsl: 0 fnx: 0x0 ver: 0x01

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x000000000.0000.00 C--- 0</td>
<td>scn 0x0000.0043236f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>0x0000.000.00000000</td>
<td>0x000000000.0000.00 ---- 0</td>
<td>fsc 0x0000.00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03</td>
<td>0x0000.000.00000000</td>
<td>0x000000000.0000.00 ---- 0</td>
<td>fsc 0x0000.00000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Dump the EMP2 block (Partial)

**block_row_dump:**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Row 0</th>
<th>@0x1f5d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tel</td>
<td>43 FB</td>
<td>--H-FL--</td>
</tr>
<tr>
<td>LB</td>
<td>0x0</td>
<td>cc: 8</td>
</tr>
<tr>
<td>Col 0</td>
<td>[3]</td>
<td>c2 4b 64</td>
</tr>
<tr>
<td>Col 1</td>
<td>[5]</td>
<td>41 4c 4c 45 4e</td>
</tr>
<tr>
<td>Col 2</td>
<td>[8]</td>
<td>53 41 4c 45 53 4d 41 4e</td>
</tr>
<tr>
<td>Col 3</td>
<td>[3]</td>
<td>c2 4d 63</td>
</tr>
<tr>
<td>Col 4</td>
<td>[7]</td>
<td>77 b5 02 14 01 01 01</td>
</tr>
<tr>
<td>Col 5</td>
<td>[2]</td>
<td>c2 11</td>
</tr>
<tr>
<td>Col 6</td>
<td>[2]</td>
<td>c2 04</td>
</tr>
<tr>
<td>Col 7</td>
<td>[2]</td>
<td>c1 1f</td>
</tr>
</tbody>
</table>

\[ALLEN\]
Dump the EMP2 block (Partial)

\texttt{tab 0, row 1, @0x1f36}
\texttt{tl: 39 fb: --H-FL-- lb: 0x0 cc: 8}
\texttt{col 0: [3] c2 50 23}
\texttt{col 1: [6] 4d 49 4c 4c 45 52 M I L L E R}
\texttt{col 2: [5] 43 4c 45 52 4b}
\texttt{col 3: [3] c2 4e 53}
\texttt{col 4: [7] 77 b6 01 17 01 01 01}
\texttt{col 5: [2] c2 0e}
\texttt{col 6: *NULL*}
\texttt{col 7: [2] c1 0b}
\texttt{end_of_block_dump}
Update ‘MILLER’ to ‘SMALL’

```sql
update emp2
set     ename = 'SMALL'
where ename = 'MILLER';

select empno, ename, job
from   emp2;
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>JOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>7499</td>
<td>ALLEN</td>
<td>SALESMAN</td>
</tr>
<tr>
<td>7934</td>
<td>SMALL</td>
<td>CLERK</td>
</tr>
</tbody>
</table>

81
Dump the EMP2 block (Partial)

**Alter system dump datafile 2 block 57810;**
*System Altered.*

Start dump data blocks tsn: 0 file#: 1 minblk 57810 maxblk 57810
buffer tsn: 0 rdba: 0x0040e1d2 (1/57810)
scn: 0x0000.00432794 seq: 0x05 flg: 0x00 tail: 0x27940605
frmt: 0x02 chkval: 0x0000 type: 0x06=trans data
Block header dump: 0x0040e1d2
Object id on Block? Y
seg.obj: 0xd076 csc: 0x00.43236f itc: 3 flg: O typ: 1 - DATA
     fsl: 0 fnx: 0x0 ver: 0x01

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x000000000.0000.00 C--- 0</td>
<td>scn 0x0000.0043236f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.07 ---- 1</td>
<td>fsc 0x0001.00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03</td>
<td>0x0000.000.00000000</td>
<td>0x000000000.0000.00 ---- 0</td>
<td>fsc 0x0000.00000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Here is the ITL with our Transaction

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x0080353.0a9e.07</td>
<td>----</td>
<td>1</td>
<td>fsc 0x0001.00000000</td>
</tr>
</tbody>
</table>

The row that was updated is still locked; fsc is 0x1 = 1 bytes.

- **Transaction ID**: Undo 4 = 4 (decimal)
- **Slot**: 2a = 42 (decimal)
- **Seq**: 12ff = 4863 (decimal)

**UBA**:  
File.block.sequence.record  
Undo block address where last change is recorded.
select segment_name
from dba_rollback_segs
where segment_id = 4;

<table>
<thead>
<tr>
<th>SEGMENT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>_SYSSMU4$</td>
</tr>
</tbody>
</table>

select header_file, header_block
from dba_segments
where segment_name = '_SYSSMU4$';

<table>
<thead>
<tr>
<th>HEADER_FILE</th>
<th>HEADER_BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>57</td>
</tr>
</tbody>
</table>
Dump the UNDO Header (Partial)  
Transaction Table (last modified blk)!

*Alter system dump datafile 2 block 57; System Altered.*

**TRN TBL:**

<table>
<thead>
<tr>
<th>index</th>
<th>state</th>
<th>cflags</th>
<th>wrap#</th>
<th>uel</th>
<th>scn</th>
<th>dba (uba)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent-xid</td>
<td>nub</td>
<td>stmt_num</td>
<td>cmt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>9</td>
<td>0x00</td>
<td>0x12ff</td>
<td>0x0001</td>
<td>0x0000.00432687</td>
<td>0x0080034e</td>
<td></td>
</tr>
<tr>
<td>0x0000.000.00000000</td>
<td>0x00000001</td>
<td>0x00000000</td>
<td>1114490213</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>9</td>
<td>0x00</td>
<td>0x12ff</td>
<td>0x0002</td>
<td>0x0000.0043269d</td>
<td>0x0080034e</td>
<td></td>
</tr>
<tr>
<td>0x0000.000.00000000</td>
<td>0x00000001</td>
<td>0x00000000</td>
<td>1114490272</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x29</td>
<td>9</td>
<td>0x00</td>
<td>0x12ff</td>
<td>0x0028</td>
<td>0x0000.004327a4</td>
<td>0x00800353</td>
<td></td>
</tr>
<tr>
<td>0x0000.000.00000000</td>
<td>0x00000001</td>
<td>0x00000000</td>
<td>1114490829</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2a</td>
<td>10</td>
<td>0x80</td>
<td>0x12ff</td>
<td>0x0002</td>
<td>0x0000.00432795</td>
<td>0x00800353</td>
<td></td>
</tr>
<tr>
<td>0x0000.000.00000000</td>
<td>0x00000001</td>
<td>0x00000000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*End dump data blocks tsn: 1 file#: 2 minblk 57 maxblk 57*
### Dump the UNDO Header (Partial)

Transaction Table:

<table>
<thead>
<tr>
<th>index</th>
<th>state</th>
<th>cflags</th>
<th>wrap#</th>
<th>uel</th>
<th>scn</th>
<th>dba</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2a</td>
<td>10</td>
<td>0x80</td>
<td>0x12ff</td>
<td>0x0002</td>
<td>0x0000.00432795</td>
<td>0x00800353</td>
</tr>
</tbody>
</table>

- **State:**
  - State 10 is Uncommitted

- **Wrap/Seq:**
  - The Wrap is 12ff

- **Slot:**
  - The Slot was 2a which was the 42\textsuperscript{nd} in the list.

- **Scn:**
  - The scn for uncommitted (ours is) or committed transactions

- **Dba:**
  - The dba of the undo which we need to look in.
The ITL again… (fyi to see UBA)

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.07</td>
<td>----</td>
<td>1</td>
<td>fsc 0x0001.00000000</td>
</tr>
</tbody>
</table>

The row that was updated is still locked; fsc is 0x1 = 1 bytes.

**Transaction ID**
- Undo 4 = 4 (decimal)
- Slot 2a = 42 (decimal)
- Seq 12ff = 4863 (decimal)

**UBA:**
- File.block.sequence.record
- Undo block address where last change is recorded.
Find the Segment & Location of RBS

```
SELECT DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(
    TO_NUMBER('00800353','XXXXXXXX')) UFILE
FROM    DUAL;

UFFILE
-------
   2

SELECT DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(
    TO_NUMBER('00800353','XXXXXXXX')) BLOCK
FROM    DUAL

BLOCK
-------
   851

Alter system dump datafile 2 block 851;
System altered.
```
Dump the UNDO Block (Partial)

UNDO BLK:

xid: 0x0004.02a.000012ff  seq: 0x9e  cnt: 0x7  irb: 0x7  icl: 0x0  flg: 0x0000

... Rec #0x7  slt: 0x2a  objn: 53366(0x0000d076)  objd: 53366  tblspc: ...

uba: 0x00800353.0a9e.04  ctl max scn: 0x0000.00432655  prv tx scn: 0x0000.00432656

txn start scn: scn: 0x0000.00432731  logon user: 0

prev brb: 8389454  prev bcl: 0
KDO undo record:
KTB Redo
op: 0x03  ver: 0x01
op: Z
KDO Op code: URP row dependencies Disabled
atypes: XA  flags: 0x00000000  bdba: 0x0040e1d2  hdba: 0x0040e1d1
itli: 2  ispac: 0  maxfr: 4863
tabn: 0  slot: 1(0x1)  flag: 0x2c  lock: 0  ckix: 0
ncol: 8  nnew: 1  size: 1

col 1: [ 6] 4d 49 4c 4c 45 52  Here’s the UNDO: MILLER
update emp2
set    ename = ‘BIG’
where ename = ‘ALLEN’;

select empno, ename, job
from   emp2;

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>JOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>7499</td>
<td>BIG</td>
<td>SALESMAN</td>
</tr>
<tr>
<td>7934</td>
<td>SMALL</td>
<td>CLERK</td>
</tr>
</tbody>
</table>
The ITL again… (fyi to see UBA)

UBA:
We’re now at Record 8.

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x000000000.0000000 C---</td>
<td>0</td>
<td>scn</td>
<td>0x0000.0043236f</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x0080353.0a9e.08 ----</td>
<td>2</td>
<td>fsc</td>
<td>0x0003.00000000</td>
</tr>
<tr>
<td>0x03</td>
<td>0x0000.000.00000000</td>
<td>0x000000000.0000000 ----</td>
<td>0</td>
<td>fsc</td>
<td>0x0000.00000000</td>
</tr>
</tbody>
</table>

Two updates for the same user uses the same ITL. Now save 3 bytes if we commit.
Dump the UNDO Block (Partial)

UNDO BLK:

xid: 0x0004.02a.000012ff  seq: 0xa9e cnt: 0x7  irb: 0x7  icl: 0x0  flg: 0x0000

* Rec #0x7  slt: 0x2a  objn: 53366(0x0000d076)  objd: 53366  tblspc: 0(0x00000000)
*-----------------------------
uba: 0x00800353.0a9e.04  ctl max scn: 0x0000.00432655  prv tx scn: 0x0000.00432656
txn start scn: 0x0000.00432731  logon user: 0
KDO undo record:
KTB Redo

col  1: [ 6]  4d 49 4c 4c 45 52  UNDO RECORD: M I L L E R
* Rec #0x8  slt: 0x2a  objn: 53366(0x0000d076)  objd: 53366  tblspc: 0(0x00000000)
*-----------------------------
KDO undo record:
KTB Redo
op: C  uba: 0x00800353.0a9e.07

col  1: [ 5]  41 4c 4c 45 4e  UNDO RECORD: A L L E N

End dump data blocks tsn: 1 file#: 2 minblk 851 maxblk 851
Now insert some records as user2

```sql
insert into emp2
select * from emp1'
;
```

14 rows created.

```sql
Alter system dump datafile 1 block 57810;
```

System altered.
Here is the ITL with our Transaction

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x00000000.0000.00 C--- 0</td>
<td>scn 0x0000.0043236f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.08 ---- 2</td>
<td>fsc 0x0003.00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03</td>
<td>0x0005.00e.0000029b</td>
<td>0x008005b2.02c9.19 ---- 14</td>
<td>fsc 0x0000.00000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transaction ID
Now there are 2 ITL’s in use.
Undo 4,5 = 4,5 (decimal)

UBA:
2 Undo Headers are used & 2 Undo blocks are referenced.

2 rows were updated on one ITL and 14 are inserted on the other.
Let’s check V$TRANSACTION & match it up to ITL (no need to dump)

```
select xidusn, xidslot, xidsqn, ubafil, ubablk, ubasqn, ubarec
from v$transaction t, v$session s
where t.ses_addr = s.saddr;
```

<table>
<thead>
<tr>
<th>XIDUSN</th>
<th>XIDSLOT</th>
<th>XIDSQN</th>
<th>UBAFIL</th>
<th>UBABLK</th>
<th>UBASQN</th>
<th>UBAREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>42</td>
<td>4863</td>
<td>2</td>
<td>851</td>
<td>2718</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>667</td>
<td>2</td>
<td>1458</td>
<td>713</td>
<td>25</td>
</tr>
</tbody>
</table>

\[4.42.4863 = 4.2a.12ff\]
\[5.14.667 = 5.e.29b\]

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.08</td>
<td>----</td>
<td>2</td>
<td>fsc 0x0003.00000000</td>
</tr>
<tr>
<td>0x03</td>
<td>0x0005.00e.0000029b</td>
<td>0x008005b2.02c9.19</td>
<td>----</td>
<td>14</td>
<td>fsc 0x0000.00000000</td>
</tr>
</tbody>
</table>
Row Level Locks

Cache Buffers

Hash Buckets

1

2

3

n

(Chain Latch)

Buffer Header

LRU Lists

LRU Write List

User Request

SGA

Shared Pool

Buffer Cache

Log Buffers

Log Buffers

DB Write

Buffer Cache

Log Buffers

DB Write

User Request
Commit EVERYTHING!

Commit; (all sessions)

Alter system dump datafile 1 block 57810;
System altered.

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x000000000.0000.00</td>
<td>C---</td>
<td>0</td>
<td>scn 0x0000.0043236f</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.08</td>
<td>----</td>
<td>2</td>
<td>fsc 0x0003.00000000</td>
</tr>
<tr>
<td>0x03</td>
<td>0x0005.00e.0000029a</td>
<td>0x008005b2.029c9.19</td>
<td>----</td>
<td>14</td>
<td>fsc 0x0000.00000000</td>
</tr>
</tbody>
</table>

Why no Change (show uncommitted)??  Delayed Block Cleanout! (Usually fast commit)
Delayed block cleanout…

Select * from emp2;
(delayed block cleanout is how redo can be generated from a select)

Alter system dump datafile 1 block 57810;
System altered.

<table>
<thead>
<tr>
<th>Itl</th>
<th>Xid</th>
<th>Uba</th>
<th>Flag</th>
<th>Lck</th>
<th>Scn/Fsc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0xffff.000.00000000</td>
<td>0x00000000.0000.00</td>
<td>C---</td>
<td>0</td>
<td>scn 0x0000.0043236f</td>
</tr>
<tr>
<td>0x02</td>
<td>0x0004.02a.000012ff</td>
<td>0x00800353.0a9e.08</td>
<td>C---</td>
<td>0</td>
<td>scn 0x0000.0043469f</td>
</tr>
<tr>
<td>0x03</td>
<td>0x0005.00e.0000029a</td>
<td>0x008005b2.02c9.19</td>
<td>C---</td>
<td>0</td>
<td>scn 0x0000.004346a3</td>
</tr>
</tbody>
</table>

All records now show as committed.
Decoding the Hot/Cold Regions (fyi only)
The percent of buffers in the hot region
A look at pointers in the LRU

Managed as FIFO

 managed as LRU/MRU

_db_percent_hot_default = 50
The percent of buffers in the hot region

Managed as FIFO

 managed as LRU/MRU

_db_percent_hot_default = 70 (too high?)
The percent of buffers in the hot region
A look at pointers in the LRU

Managed as FIFO

Managed as LRU/MRU

If Touch Count > 2 when Oracle is looking for free buffer or doing DBWR then pointer to MRU (hot).
Altering the Hot/Cold LRU
Really advanced tuning!

1. \_db\_percent\_hot\_default (50) – The percent of buffers in the hot region.

2. \_db\_aging\_touch\_time* (3) – Seconds that must pass to increment touch count again. (Higher - less LRU movement)

3. \_db\_aging\_hot\_criteria (2) – Threshold to move a buffer to the MRU (hot) end of LRU chain.

4. \_db\_aging\_stay\_count (0) – **Touch count reset to this when moved to MRU (hot) end. Set=0 even if it was 200 previously!

5. \_db\_aging\_cool\_count (1) – Touch count reset to this when moved to LRU (cold) end. Set=1 even if it was 200 previously!

*Setting parameter 1 (above) lower, we increase hanging on to older buffers and setting it higher will cause a flush sooner. (*Error in description)*
6. `_db_aging_freeze_cr` (FALSE) – Setting this to TRUE will make cr (consistent read) buffers too cold to keep in the cache.

7. `_db_percent_hot_keep (0)` – Percent of keep buffers considered hot (in hot region). Old LRU algorithm!

8. `_db_percent_hot_recycle (0)` – Percent of recycle buffers considered hot (in hot region). Old LRU algorithm!

**If `_db_aging_stay_count => _db_aging_hot_criteria` then touch count is set to ½ it’s current count instead of setting it to the `_db_aging_stay_count` when moved to the hot end of LRU.**

- FTS, FFIS *(multi-block)* are put on the cold end of the LRU.
Queries that may help you find solutions your problems...FYI ONLY
select object_name, statistic_name, value
from v$segment_statistics
where value > 100000
order by value;

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>STATISTIC_NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERS</td>
<td>space allocated</td>
<td>96551</td>
</tr>
<tr>
<td>ORDERS</td>
<td>space allocated</td>
<td>134181</td>
</tr>
<tr>
<td>ORDERS</td>
<td>logical reads</td>
<td>140976</td>
</tr>
<tr>
<td>ORDER_LINES</td>
<td>db block changes</td>
<td>183600</td>
</tr>
</tbody>
</table>
Update a row as one user and then update the same row as another user (waiting):

```sql
select event, total_waits, total_timeouts
from v$system_event
where event like '%enq%'
```

<table>
<thead>
<tr>
<th>EVENT</th>
<th>TOTAL_WAITS</th>
<th>TOTAL_TIMEOUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>enq: PR - contention</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>enq: RO - fast object reuse</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>enq: TX - row lock contention</td>
<td>337</td>
<td>336</td>
</tr>
<tr>
<td>enq: WF - contention</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>enq: JS - slave enq get lock1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Great V$ - V$SESSION in 10g has it all!

```sql
select username, sid, event, seq#, seconds_in_wait, wait_time, p1, p2, p3, st
from V$SESSION
where NOT(event like 'SQL%')
and NOT(event like '%message%')
and NOT(event like '%timer%')
and NOT(event like '%pipe get%')
order by wait_time desc, event;
```

V$SESSION EVEN HAS: row_wait_obj#, row_wait_file#, row_wait_block#, row_wait_row#
### Great V$ - V$SESSION has it all!

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>SID</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEQ#</th>
<th>SECONDS_IN_WAIT</th>
<th>WAIT_TIME</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>159 Queue Monitor Wait</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WAITING</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS</td>
<td>159 enq: TX - row lock contention</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WAITING</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>162</td>
<td>jobq slave wait</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WAITING</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>147</td>
<td>wakeup time manager</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WAITING</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### x$ksqst – Kernel service enqueue statistics by type (no waits yet)

```
select ksqsttyp type, ksqstsgt gets, ksqstwat waits
from x$ksqst
where ksqstsgt > 0
order by waits , gets
```

<table>
<thead>
<tr>
<th>TYPE</th>
<th>GETS</th>
<th>WAITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td>6313</td>
<td>0</td>
</tr>
<tr>
<td>HW</td>
<td>9091</td>
<td>0</td>
</tr>
<tr>
<td>TX</td>
<td>55309</td>
<td>0</td>
</tr>
<tr>
<td>TM</td>
<td>101973</td>
<td>0</td>
</tr>
<tr>
<td>CF</td>
<td>132720</td>
<td>0</td>
</tr>
</tbody>
</table>
x$bh – Cache buffers chains issues

- To identify a heavily accessed buffer chain look at the latch statistics for cache buffers chains latch under V$LATCH_CHILDREN and match this to X$BH:

  ```sql
  select dbarfil, dbablk, class, state
  from x$bh
  where bladdr = '<latch address from v$latch_children>'
  ```
X$ - Finding Specific waits

Buffer Busy Waits or Write Complete Waits Events:

```
SELECT /*+ ordered */ sid, event, owner, segment_name, segment_type, p1, p2, p3
FROM v$session_wait sw, dba_extents de
WHERE de.file_id = sw.p1
AND sw.p2 between de.block_id and de.block_id+de.blocks - 1
AND (event = 'buffer busy waits'
    OR event = 'write complete waits')
AND p1 IS NOT null
ORDER BY event, sid;
```
col name for a20
col p1 for a10
col p2 for a10
col p3 for a10

select event#, name, parameter1 p1, parameter2 p2, parameter3 p3
from v$event_name
where name in ('buffer busy waits', 'write complete waits')

<table>
<thead>
<tr>
<th>EVENT#</th>
<th>NAME</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>143</td>
<td>write complete waits</td>
<td>file#</td>
<td>block#</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>buffer busy waits</td>
<td>file#</td>
<td>block#</td>
<td>id</td>
</tr>
</tbody>
</table>
Combining Trace and Wait Events!

- Querying the V$SESSION_WAIT table gives a point in time view of the waits
- To record all the waits use
  - `Alter session set events '10046 trace name context forever, level 12';`
- In the trace file you’ll see entries such:
  - `WAIT #2: nam='db file sequential read'
    ela= 0 p1=1 p2=1135 p3=1`
    - #2 - is the SQL that caused the wait
    - P1 - file#, P2 - block#, p3 - number of block read
Lots of latch free waits in this one…

WAIT #2: nam='latch free' ela= 0 p1=-2147423252 p2=105 p3=0
WAIT #2: nam='latch free' ela= 0 p1=-2147423252 p2=105 p3=1
WAIT #2: nam='latch free' ela= 0 p1=-1088472332 p2=106 p3=0
WAIT #2: nam='latch free' ela= 0 p1=-2147423252 p2=105 p3=0
WAIT #2: nam='latch free' ela= 0 p1=-2147423252 p2=105 p3=1
Checking this event ...

```sql
select event#, name, parameter1 p1, parameter2 p2, parameter3 p3
from v$event_name
where name in ('latch free');
```

<table>
<thead>
<tr>
<th>EVENT#</th>
<th>NAME</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>latch free</td>
<td>address</td>
<td>number</td>
<td>tries</td>
</tr>
</tbody>
</table>
select sum(pct_bufgets) percent
from (select rank() over ( order by buffer_gets desc ) as rank_bufgets,
to_char(100 * ratio_to_report(buffer_gets) over (pct_bufgets)
from v$sqlarea )
where rank_bufgets < 11;

PERCENT
---------
  97.07
If Time Permits… the Future!
Directly Addressable Memory

![Bar chart showing memory capacity for 8-bit, 16-bit, 32-bit, and 64-bit systems.](chart.png)
64-Bit advancement of Directly addressable memory

<table>
<thead>
<tr>
<th>Address Direct</th>
<th>Indirect/Extended</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Bit:</td>
<td>16</td>
</tr>
<tr>
<td>8 Bit:</td>
<td>256</td>
</tr>
<tr>
<td>16 Bit:</td>
<td>65,536</td>
</tr>
<tr>
<td>32 Bit:</td>
<td>4,294,967,296</td>
</tr>
<tr>
<td>64 Bit:</td>
<td>18,446,744,073,709,551,616</td>
</tr>
</tbody>
</table>

When the hardware physically implements the theoretical possibilities of 64-Bit, things will dramatically change.... ...moving from 32 bit to 64 bit will be like moving from 4 bit to 32 bit or like moving from 1971 to 2000 overnight.
What if directly addressable memory was pennies?

16 – Bit (65,536 bytes)

4 – Bit (16 bytes)

8 – Bit (256 bytes)
32 – Bit
(4,294,967,296 bytes)

Here is the square from the last slide for 16-bit (a bit smaller)
1 exa-penny (1,000,067,088,384,000,000 pennies) would form a 27,300 square foot cube...

In comparison, Mt. Everest at 29,000 feet is only 1,700 feet taller
64 – Bit
(18,446,744,073,709,551,616 bytes)

16 exa-pennies would easily dwarf even Mt. Everest!
8 Exabytes:
Look what fits in one 10g Database!

- 1000 Internets (8P each)
- 400,000 Libraries of Congress (20T each and 17-18 million books in each)
- 8 Billion Movies on CD (1 G each)
- 8 Billion Pickup Trucks of Documents (1G each)
- 1 Mount Everest filled with Documents (approx.)
- Every piece of data produced each year (2E-5.6E)
### Row Level Locks

<table>
<thead>
<tr>
<th>Cache Buffers</th>
<th>Hash Buckets</th>
<th>Chain Latch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

- **User Request**
- **Cache Buffers**
- **Hash Buckets**
- **Chain Latch**
- **Buffer Cache**
- **Shared Pool**
- **Log Buffers**
- **LRU Lists**
- **LRU Write List**
- **DB Write**
<table>
<thead>
<tr>
<th>Version</th>
<th>Doc.</th>
<th>Undoc.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>111</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>7</td>
<td>117</td>
<td>68</td>
<td>185</td>
</tr>
<tr>
<td>8.0</td>
<td>193</td>
<td>119</td>
<td>312</td>
</tr>
<tr>
<td>8.1</td>
<td>203</td>
<td>301</td>
<td>504</td>
</tr>
<tr>
<td>9.0</td>
<td>251</td>
<td>436</td>
<td>687</td>
</tr>
<tr>
<td>9.2</td>
<td>257</td>
<td>540</td>
<td>797</td>
</tr>
<tr>
<td>10.1.0.2</td>
<td>255</td>
<td>911 (+69%)</td>
<td>1166 (+42%)</td>
</tr>
</tbody>
</table>
We Covered:

• Terminology
• What you’re Waiting on
• Block Dumps & Block Information
• Transactions moving through Oracle
  – Buffer Cache
  – ITL & UNDO tie
  – Delayed block cleanout
For More Information

- www.tusc.com
- Oracle9i Performance Tuning Tips & Techniques; Richard J. Niemiec; Oracle Press (May 2003)

“If you are going through hell, keep going” - Churchill
“We make a Living by what we get; We make a Life by what we give.”
References

• *Oracle9i Performance Tuning Tips & Techniques*, Rich Niemiec
• Block Level Reading Tool is from Terlingua Software
• The Machinations of Oracle, Scott Marin, Terlingua software
• *All about Oracle’s touch count algorithm*, Craig Schallabamer
• Mark Bobak, Kevin Gilpin, Jonathon Lewis, Metalink Notes
• *EM Grid Control 10g*; otn.oracle.com, Oracle Corporation
• *Oracle Enterprise Manager 10g: Making the Grid a Reality*; Jay Rossiter, Oracle Corporation
• [http://www.kokogiak.com/megapenny](http://www.kokogiak.com/megapenny)
• Oracle 10g documentation
• *Special thanks to Steve Adams, Brad Brown, Kevin Gilpin, and Joe Trezzo.*
• *Oracle OLN Advanced Tuning Class*, Oracle Corporation
• [www.lookuptables.com](http://www.lookuptables.com)
• Scaling Oracle8i by James Morle
• Reading a block dump, TUSC
• [http://www.jlcomp.demon.co.uk/](http://www.jlcomp.demon.co.uk/)
TUSC Services

- **Oracle Technical Solutions**
  - Full-Life Cycle Development Projects
  - Enterprise Architecture
  - Database Services

- **Oracle Application Solutions**
  - Oracle Applications Implementations/Upgrades
  - Oracle Applications Tuning

- **Managed Services**
  - 24x7x365 Remote Monitoring & Management
  - Functional & Technical Support

- **Training & Mentoring**

- **Oracle Authorized Reseller**
Copyright Information

- Neither TUSC, Oracle, NYOUG, nor the author guarantee this document to be error-free. Please provide comments/questions to rich@tusc.com.
- TUSC © 2005. This document cannot be reproduced or used without expressed written consent from an officer of TUSC, but NYOUG may reproduce or copy this for conference use.

Contact Information

Rich Niemiec: rich@tusc.com
www.tusc.com