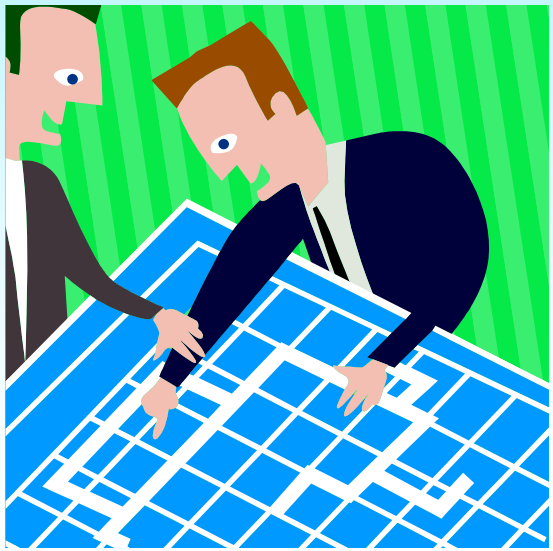
OracleWorld 2003
EOUG User2User day

Freelists vs ASSM in Oracle9i

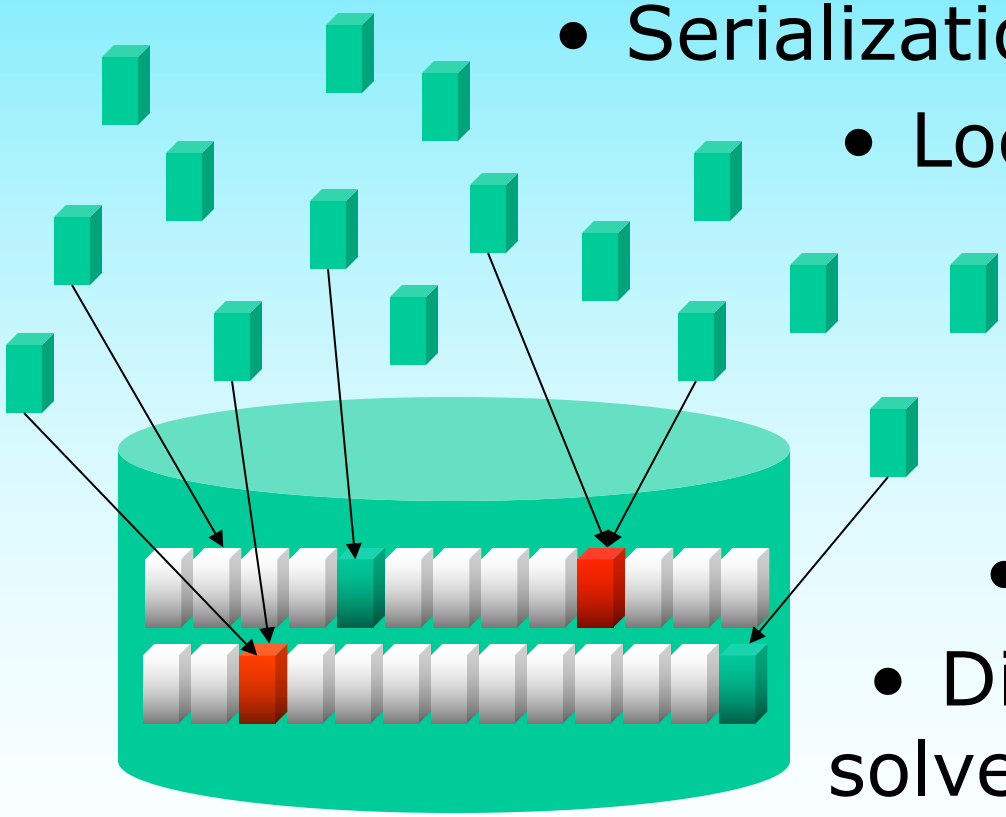
Tanel Poder
independent technology consultant
http://integrid.info

Agenda

- High concurrency environment issues
- Oracle storage & free space management
- Freelist Segment Management internals
- Automatic Segment Space Management internals
- FLM vs ASSM Comparision
- Converting to ASSM
- Conclusion
- Questions



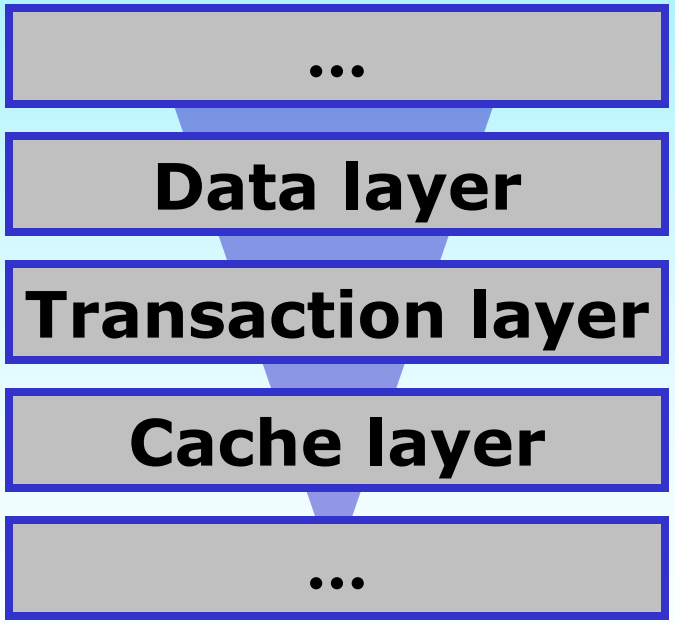
High Concurrency Environments



- Main performance problems
- Serialization vs Corruption
- Locking & Latching
- Freelists
- Treelists
- Hashing
- Load balancing
- Different methods solve different issues

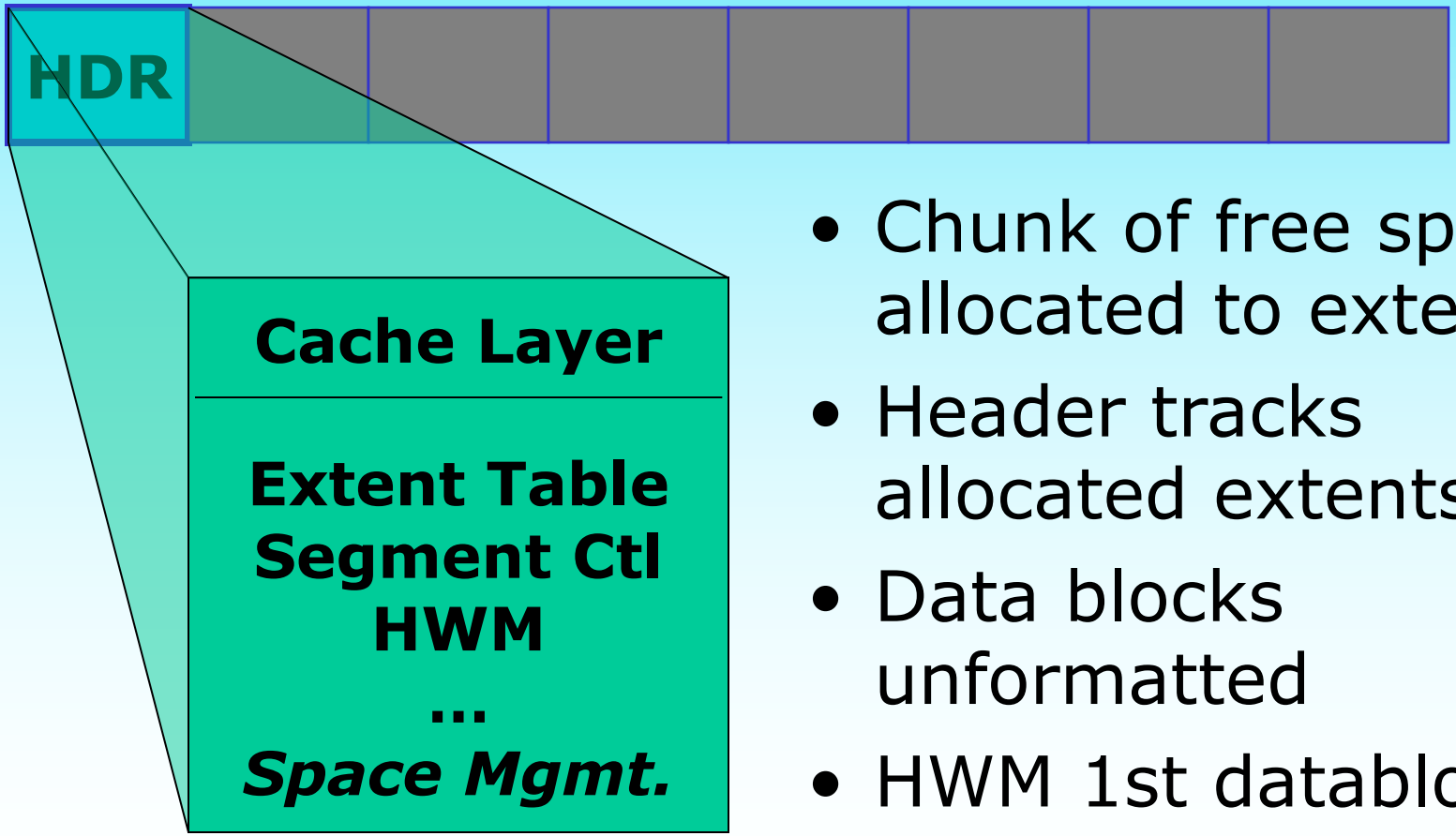
Oracle Data Storage

- Cache Layer (KC)
  - Organizes data into Oracle datablocks
  - Manages buffer cache, concurrency control
  - Does redo logging
- Transaction Layer (KT)
  - Generates undo & rollback
  - Read consistency and ITL
  - Does extent allocation
  - Manages segment space
  - PCTFREE, Freelist & ASSM



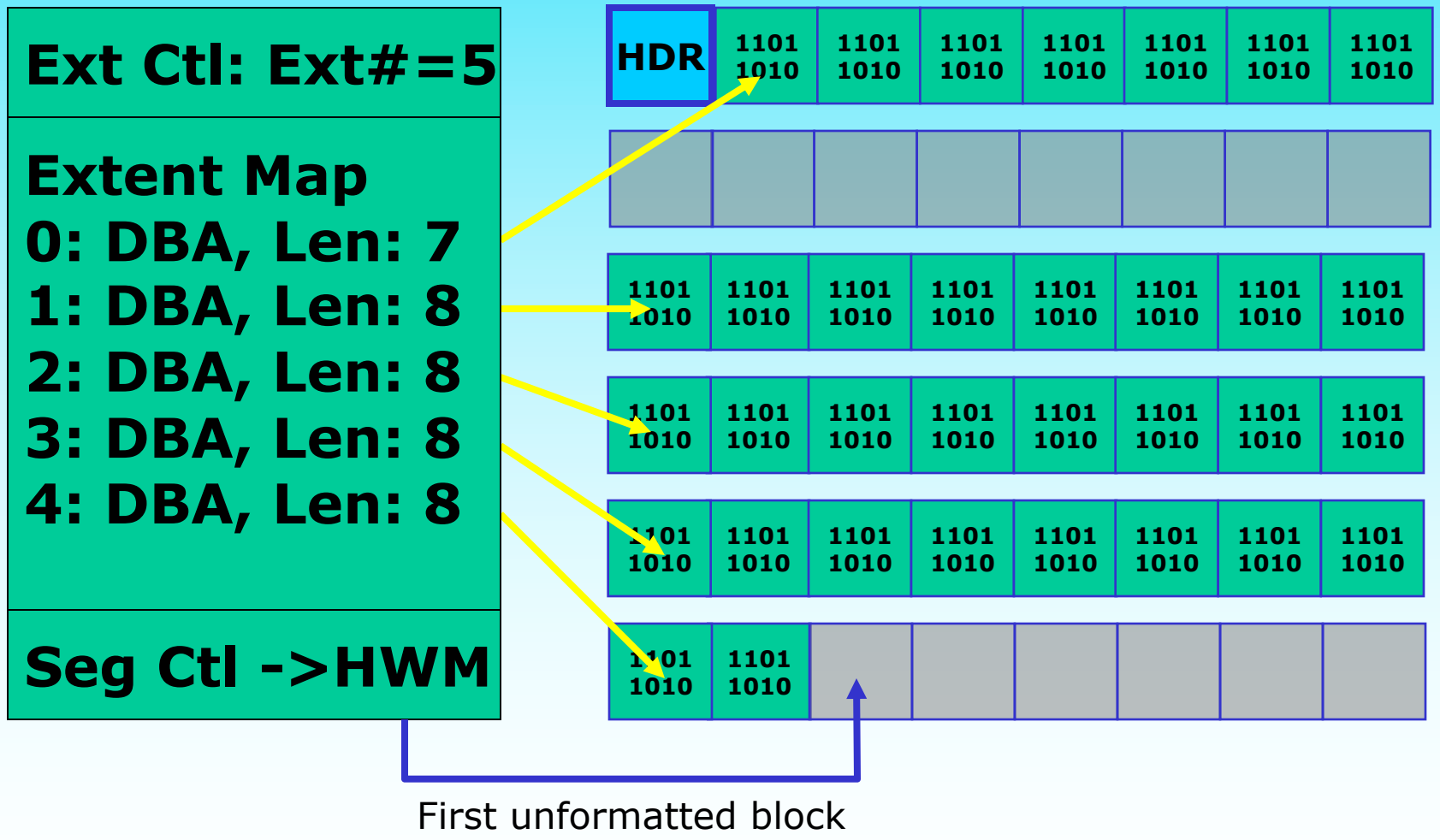
Freelist Managed Segment

CREATE TABLE T1 (col1 datatype) ;



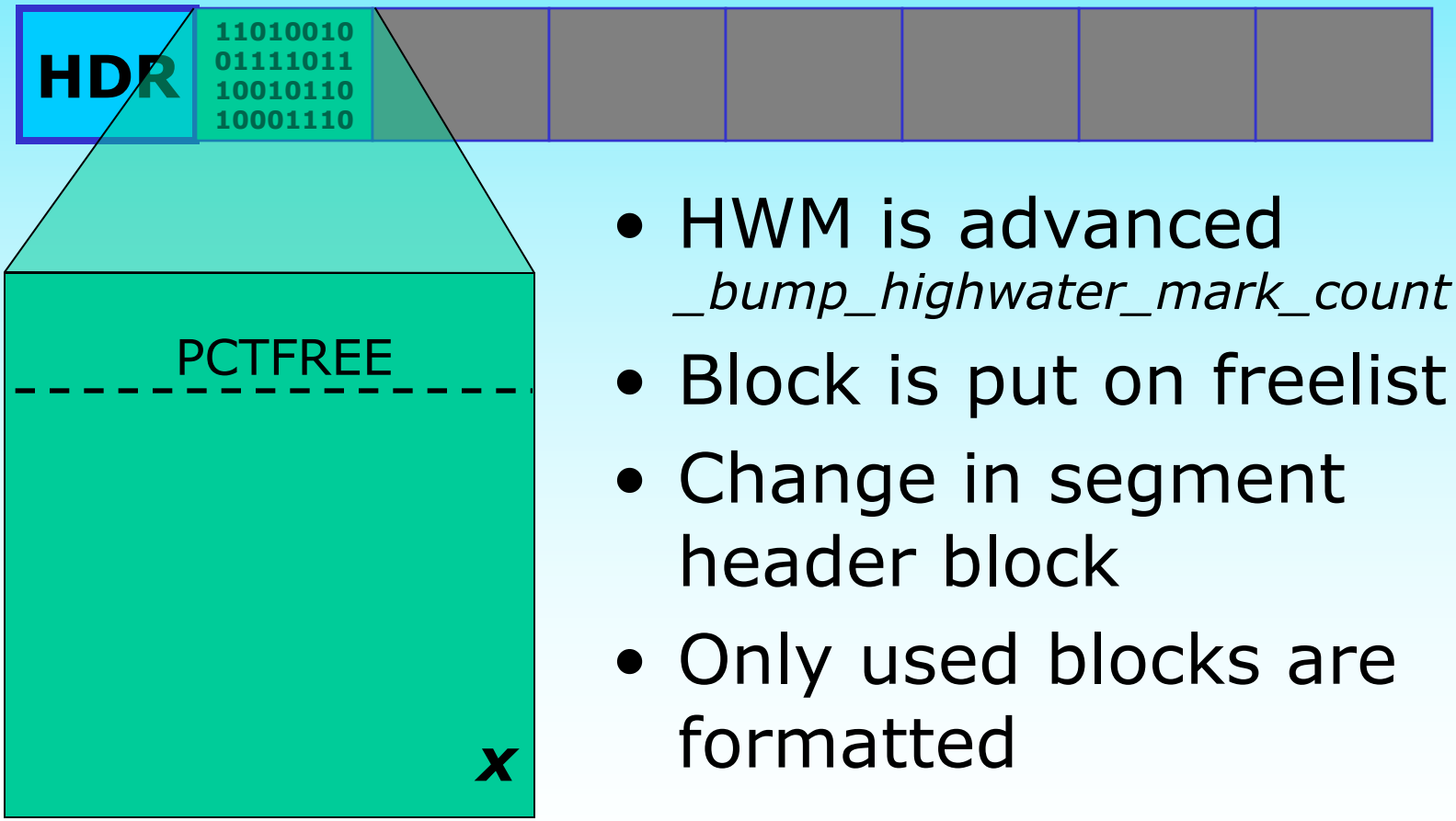
- Chunk of free space allocated to extent
- Header tracks allocated extents
- Data blocks unformatted
- HWM 1st datablock

Freelist Extent Management



FLM: First insert into segment

INSERT INTO T1 VALUES ('x') ;



- HWM is advanced *_bump_highwater_mark_count*
- Block is put on freelist
- Change in segment header block
- Only used blocks are formatted

Freelist Usage

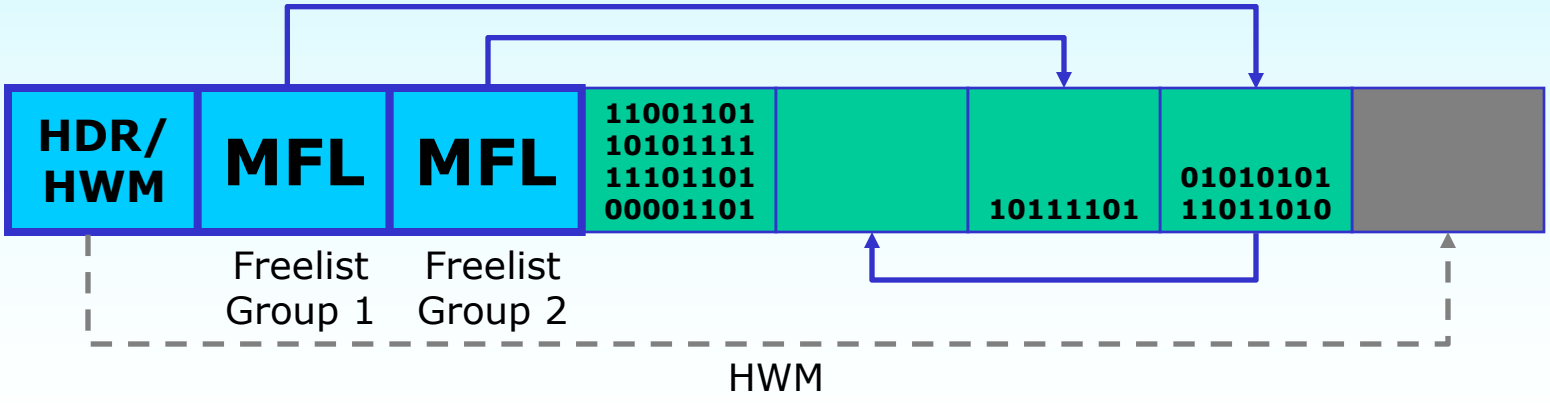
- Freelist is a data structure for keeping track of blocks candidates for inserts
- Is a Last-in First-out type linked list
  - No space overhead
- Gets new free blocks by bumping up HWM
  - HWM can be lower than formatted blocks until committed in case of direct load insert
- When an insert would cause a block to be filled over PCTFREE and block is already over PCTUSED, the block is unlinked

Three Types of Freelists

- Segment Freelist or Master Freelist (MFL)
  - The default, also called common pool
- Process Freelist (PFL)
  - Is created with FREELISTS clause
- Transaction Freelist (TFL)
  - Used implicitly when DML reduces block space utilization under PCTUSED
- Every freelist Group uses one extra data block after segment header
  - Consists of MFL, PFLs and TFLs

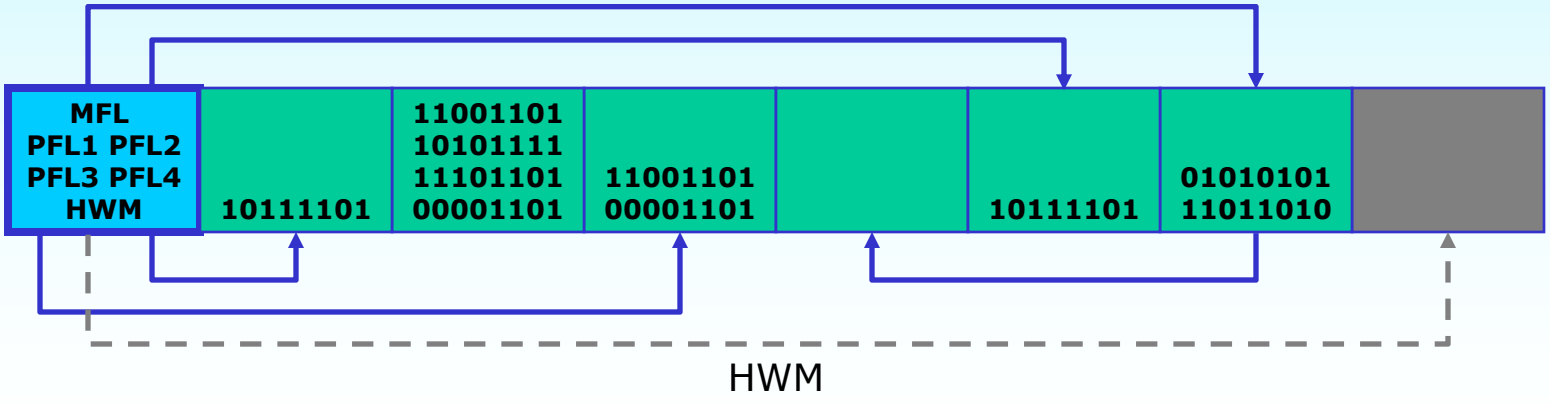
Master Free List (MFL)

- Created with every segment
- Common pool for free blocks for everyone
- All freelists reside in segment header or in special blocks in case of FREELIST GROUPS
- One MFL per freelist group + one remains in segment header (mostly unused)



Process Free List (PFL)

- Created with FREELISTS clause
- Free block pool serving group of processes
  - Spreads concurrent insert operations using PID
- Max number determined by block size
  - 99 for 8k block (internally one more is stored for MFL)

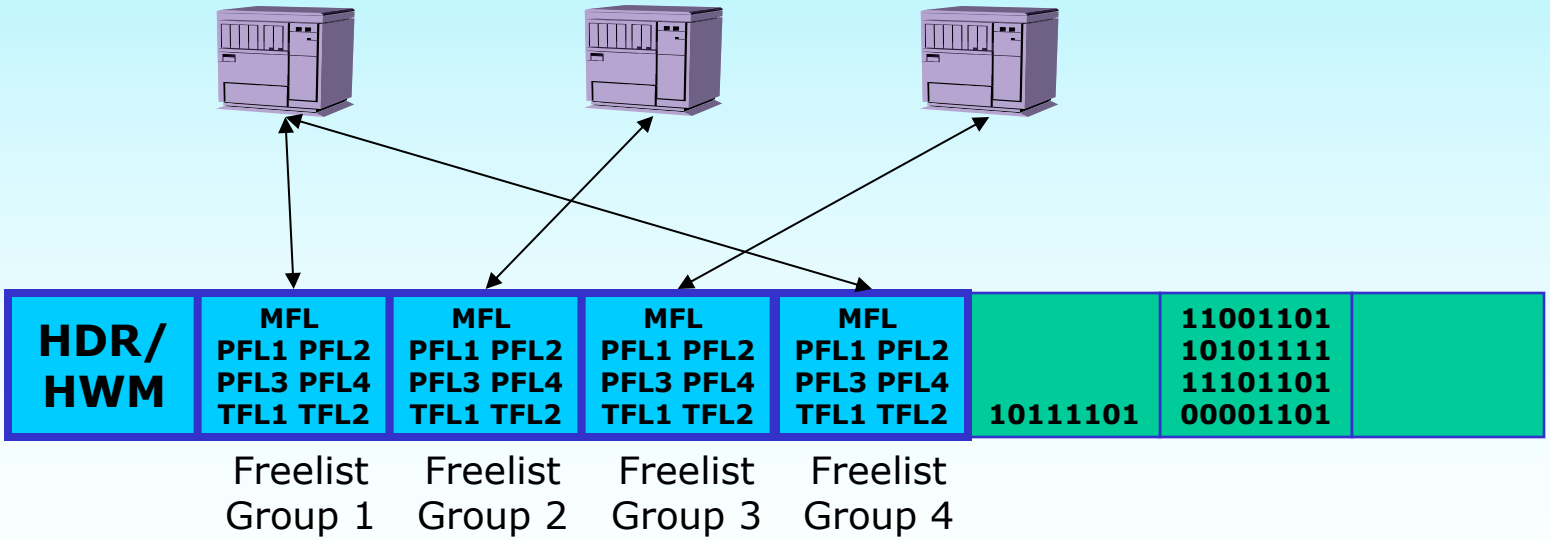


Transaction Free List (TFL)

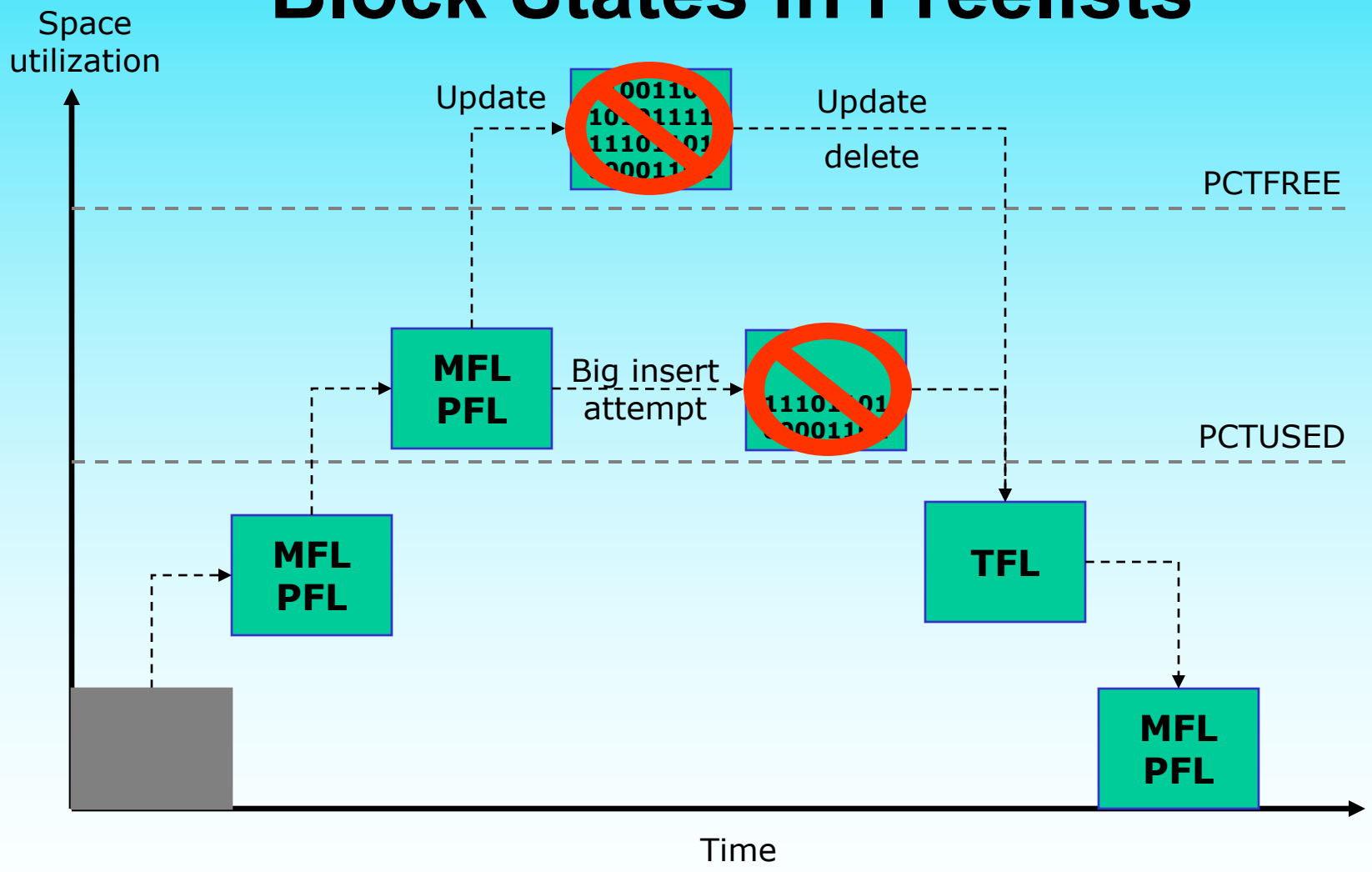
- Is only used when a delete or update operation reduces block space utilization under PCTUSED
- Freelist is only accessible to transaction which caused the transition
- After commit, the block remains in TFL
  - Is not used for any inserts since TFL is tied to specific transaction
  - Until all other freelists in current freelist group are empty - the blocks are moved to MFL or PFL

Freelist Groups

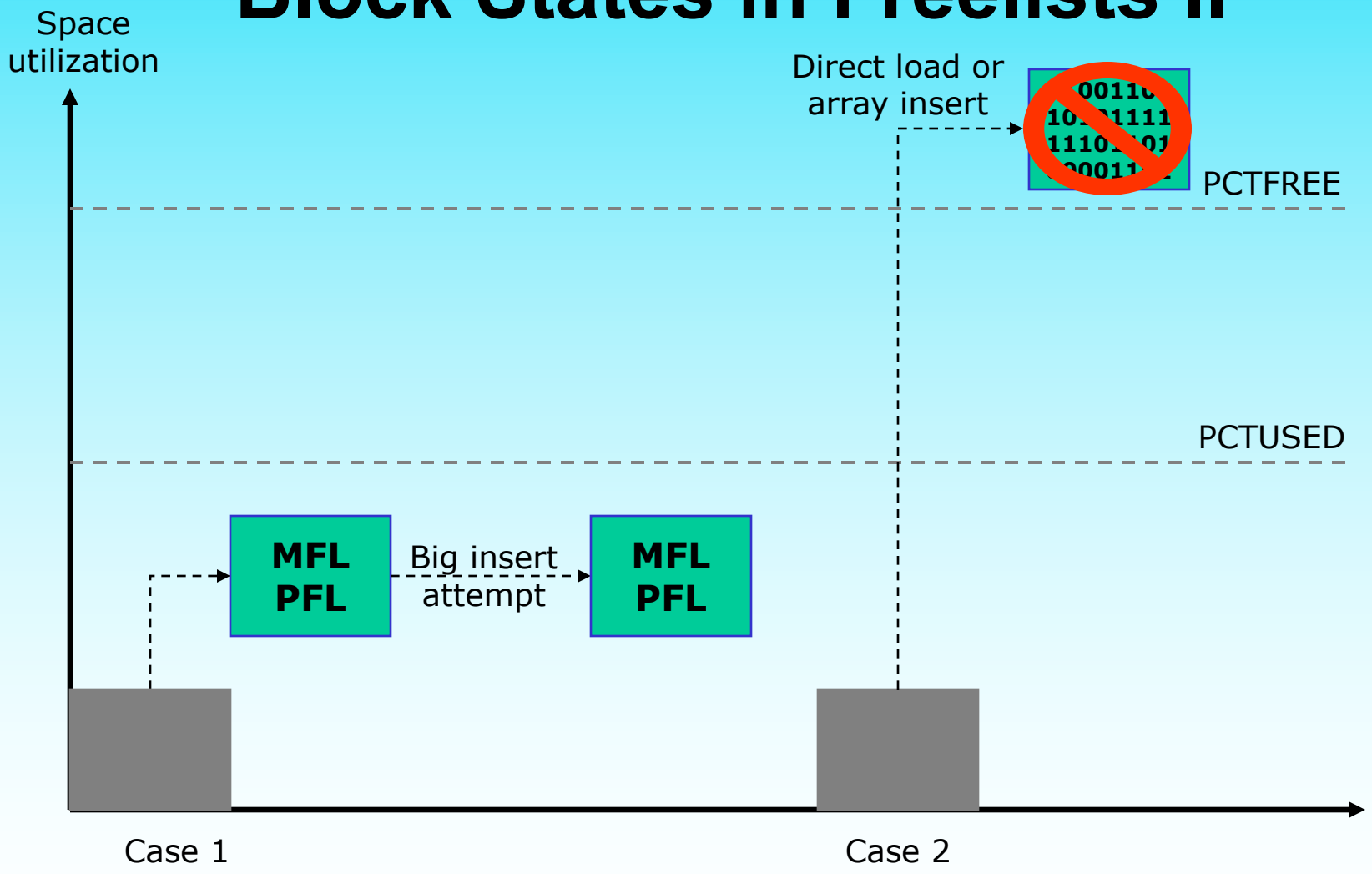
- To reduce contention on segment header
- Especially useful in OPS and RAC, when sharing one buffer would result in excessive pinging or GC traffic



Block States in Freelists



Block States in Freelists II



Freelist search stages

1) Uncommitted TFL (for current transaction)
2) Search PFL & Use if found
3) Search MFL & Move to PFL if found
4) Search Committed TFL & Move to MFL
5) Search Common pool (MFL in seg. header)
6) Bump HWM & Move to PFL
7) Allocate extent
8) Extend datafile
9) Error

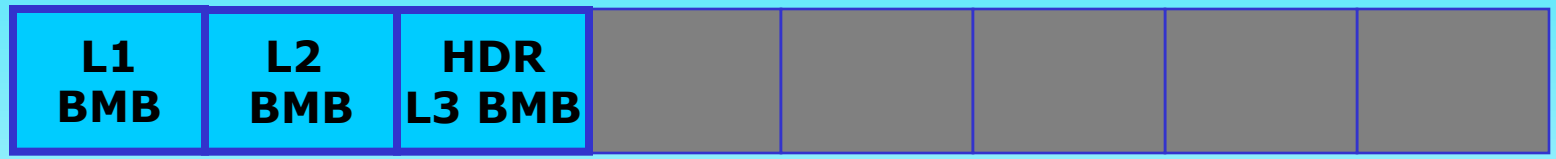
Freelist Search parameters

- Every freelist block traversed has to be read in order to get address of next block
- *_walk_insert_treshold* (default 5)
  - Freelist blocks to scan before turning to higher level list or bump HWM (if walking on TFL, PFL and MFL are searched next)
- *_release_insert_threshold* (default 5)
  - How many unsuitable blocks to unlink from freelist before bump HWM

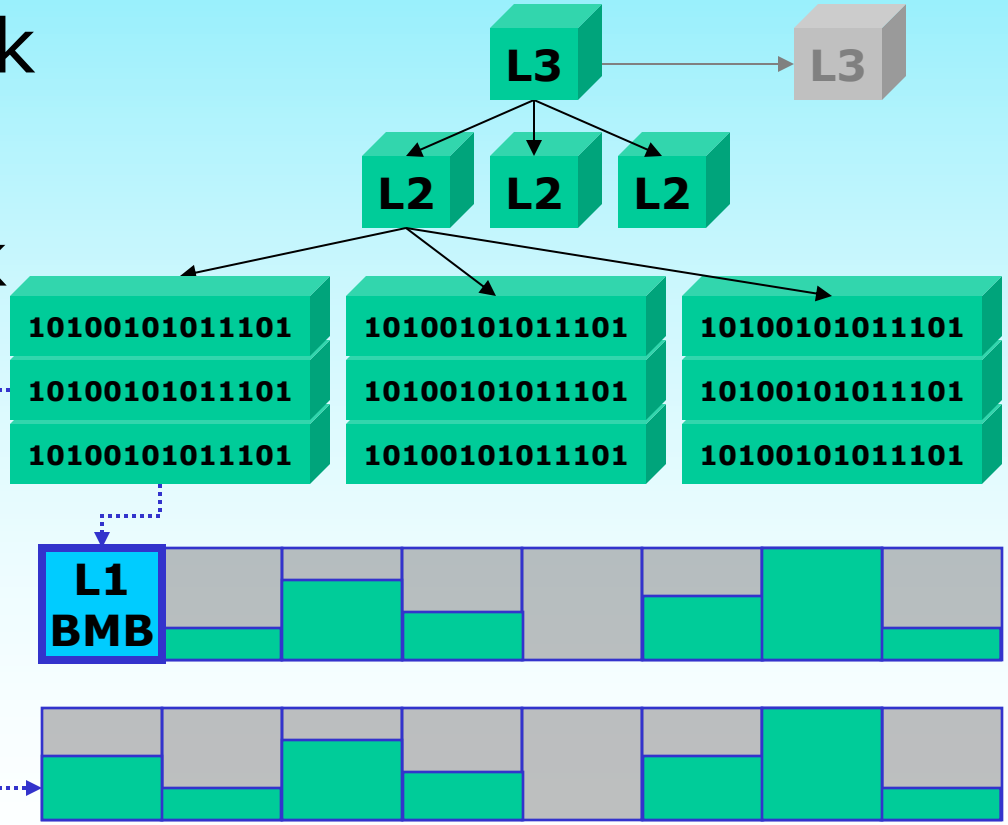
Automatic Segment Space Mgmt.

- ASSM free space structure is somewhat similar to a B-tree index structure
- Tree traversing is used for getting to block utilization information
- ASSM tree is only 3 levels high
  - Root, branch and leaf nodes
- Every datablocks "freeness" is represented using few bits in leaf nodes
- Free space searching is faster but space overhead is greater

ASSM Segment

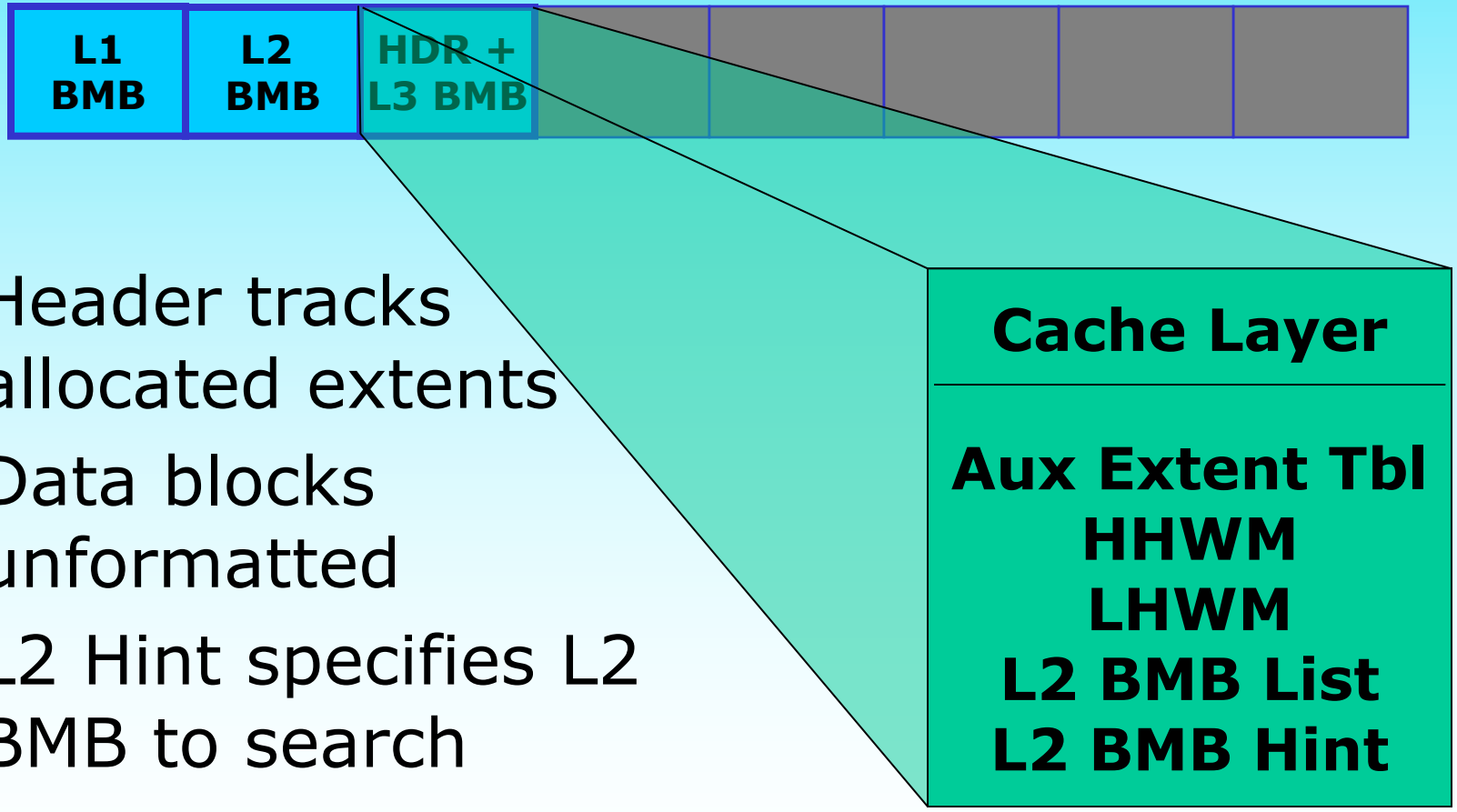


- BMB=Bitmap Block
- L3 = Root Block
- L2 = Branch Block
- L1 = Leaf BMB
- L3 can reside in segment header
- L1 BMB is always first in extent



ASSM Segment Header

CREATE TABLE T1 (col1 datatype) ;



- Header tracks allocated extents
- Data blocks unformatted
- L2 Hint specifies L2 BMB to search

ASSM: First insert into segment

INSERT INTO T1 VALUES ('x') ;

L1 Bitmap statuses:

0000 Unformatted block
0001 Block logically full
0010 <25% free space
0011 25% - 50% free
0100 50% - 75% free
0101 >75% free space

- HHWM is advanced
- Up to 16 blocks are formatted at once
- Change in segment header block
- Most activity is spread to L1 and L2 blocks

Level 1 BMB

- Level 1 BMBs indicate the “freeness” of blocks in DBA range using bitset vector
- DBA range represents contiguous set of blocks within an extent
- From 16-1024 DBA ranges per L1 BMB
- With smaller segments the relative amount of L1 BMBs is bigger to maintain concurrency benefits (one L1 for 16 blocks)
- L1 BMB is the smallest unit of space which has affinity for an instance

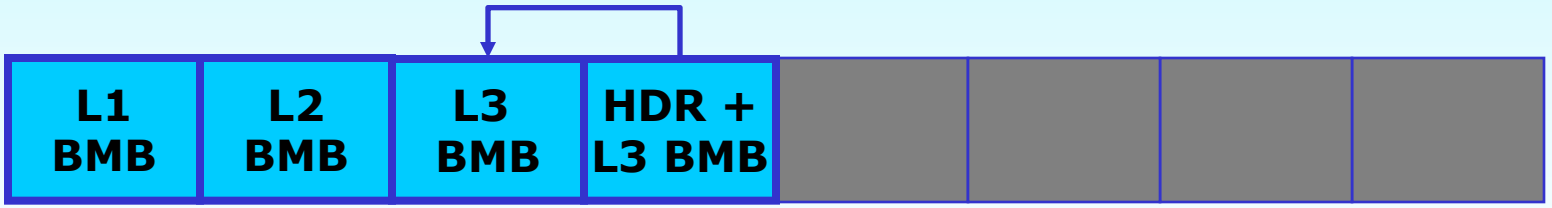
Level 2 BMB

- Contains search hint for first L1 BMB
- Count L1 BMBs with free status helps to skip L2 blocks in space search
- L1 DBA Array:
  - L1 Data Block Address
  - Instance it is mapped to (can be dynamically changed)
  - Maximum freeness in any block
    Statuses from 1-6, from unformatted to full

Big extent size

Level 3 BMB

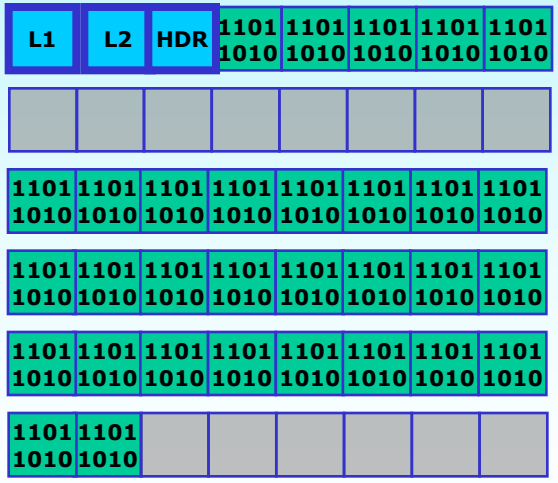
- Reside in segment header
- Organized as linked list
- Contain pointers to L2 BMBs
- In case of insufficient space in header, separate L3 BMBs are created
  - Original L2 pointers remain in segment header



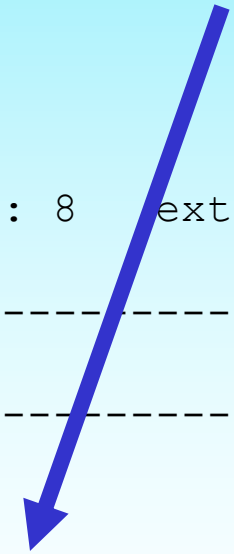
Big tablespace size

ASSM Block Formatting

```
SQL> create tablespace ts datafile 'ts.dbf' size 1m
     extent management local uniform size 64k
     segment space management auto;

SQL> create table t (a number) tablespace ts;

SQL> select file_id, block_id, blocks from
     dba_extents where segment_name  = 'T';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         9          9          8

SQL> insert into t values (1);

SQL> alter system dump datafile 9
     block min 9 block max 17;
```



```
frmt: 0x02 chkval: 0x0000 type: 0x20=FIRST LEVEL BITMAP BLOCK
Dump of First Level Bitmap Block
   --------------------------------
   nbits : 4 nranges: 1          parent dba:  0x0240000a  poffset: 0
   unformatted: 0       total: 8          first useful block: 3
   owning instance : 1
   instance ownership changed at 10/10/2003 20:43:55
   Last successful Search 10/10/2003 20:43:55
   Freeness Status:  nf1 0      nf2 0      nf3 0      nf4 5

   First free datablock : 3
   Bitmap block lock opcode 0
   Locker xid:      :  0x0000.000.00000000
      Highwater::  0x02400011  ext#: 0      blk#: 8   ext size: 8

  ------------------------------------------------------
  DBA Ranges :
  ------------------------------------------------------
   0x02400009  Length: 8      Offset: 0

   0:Metadata      1:Metadata      2:Metadata      3:75-100% free
   4:75-100% free  5:75-100% free  6:75-100% free  7:75-100% free
  ------------------------------------------------------
```



L2 Bitmap Block

```
insert into t select 1 from sys.obj$ where rownum <= 3400;

3400 rows created.

frmt: 0x02 chkval: 0x0000 type: 0x21=SECOND LEVEL BITMAP BLOCK
Dump of Second Level Bitmap Block
   number: 2        nfree: 1        ffree: 1        pdba:  0x0240000b
  opcode:0
 xid:
  L1 Ranges :
  -------------------------------------------------------
   0x02400009  Free: 1 Inst: 1
   0x02400019  Free: 5 Inst: 1

  -------------------------------------------------------
```

Maximum freeness

L1 Bitmap statuses:

0000 Unformatted block
0001 Block logically full
0010 <25% free space
0011 25% - 50% free
0100 50% - 75% free
0101 >75% free space

FS1=0010 FS2=0011
FS3=0100 FS4=0101

L3 Bitmap Block

```
frmt: 0x02 chkval: 0x0000 type: 0x22=THIRD LEVEL BITMAP BLOCK
Dump of Third Level Bitmap Block
   number: 107   , next :     0x00000000
  L2 Ranges :
  ----------------------------------------------------
     0x07112f1e
     0x071154c6     0x07117a6e     0x0711a016     0x0711c5be
     0x0711eb66     0x0712110e     0x071236b6     0x07125c5e
     . . .
```

- *number* shows number of L2 entries in current L3 block
- *next* references next L3 block in list, last block if zero
- is referenecd by *First Level 3 BMB* in segment header

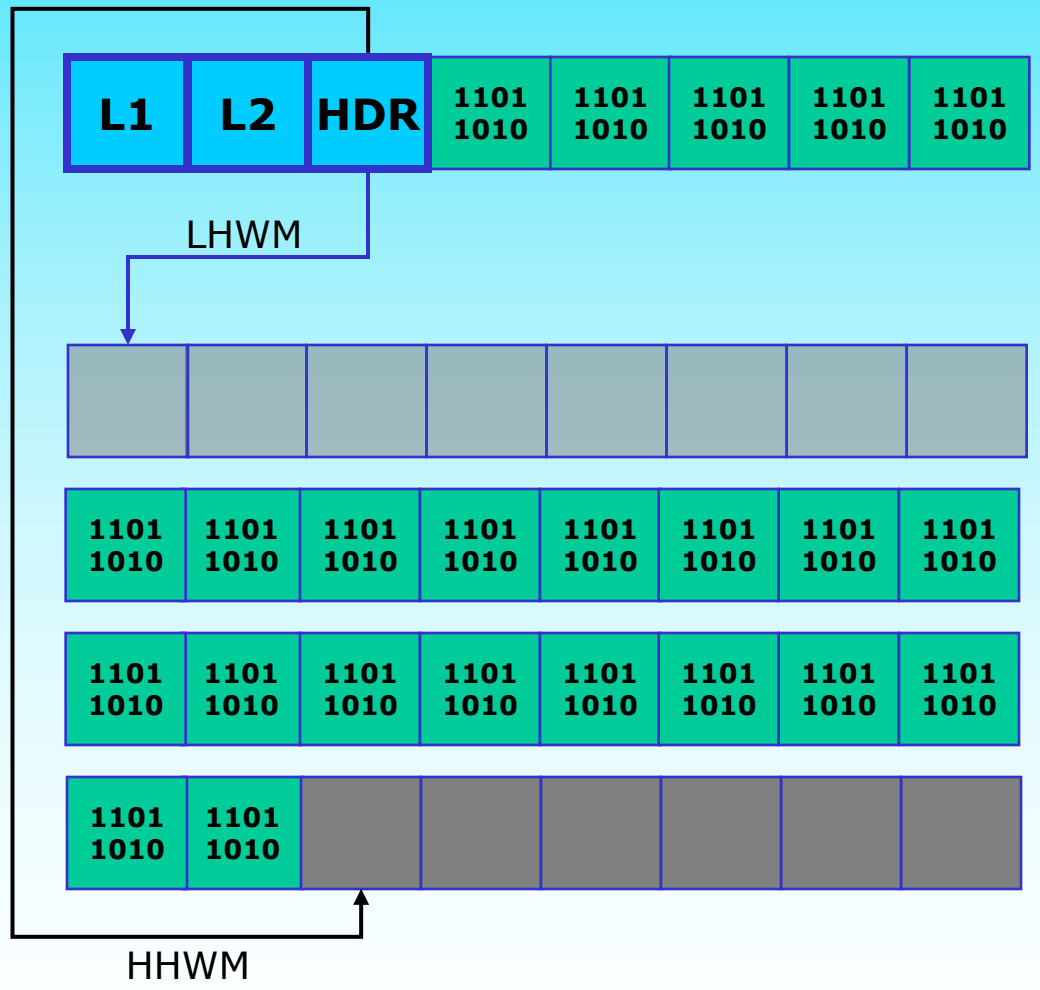
Freelist vs ASSM Datablock

FREELIST	ASSM
fnx - next block in freelist	bdba - L1 BMB address
fsl - free space lock	brn - DBA range number opcode
none	inc - incarnation of block (if HWM is pulled back)

```
Block header dump:  0x0240000e
 Object id on Block? Y
 seg/obj: 0x18a9  csc: 0x00.126b4  itc: 2  flg: E  typ: 1 - DATA
     brn: 0  bdba: 0x2400009 ver: 0x01
     inc: 0  exflg: 0

 Itl           Xid                  Uba         Flag  Lck        Scn/Fsc
0x01   0x0000.000.00000000  0x00000000.0000.00  ----    0  fsc 0x0000.00000000
0x02   0x0000.000.00000000  0x00000000.0000.00  ----    0  fsc 0x0000.00000000
```

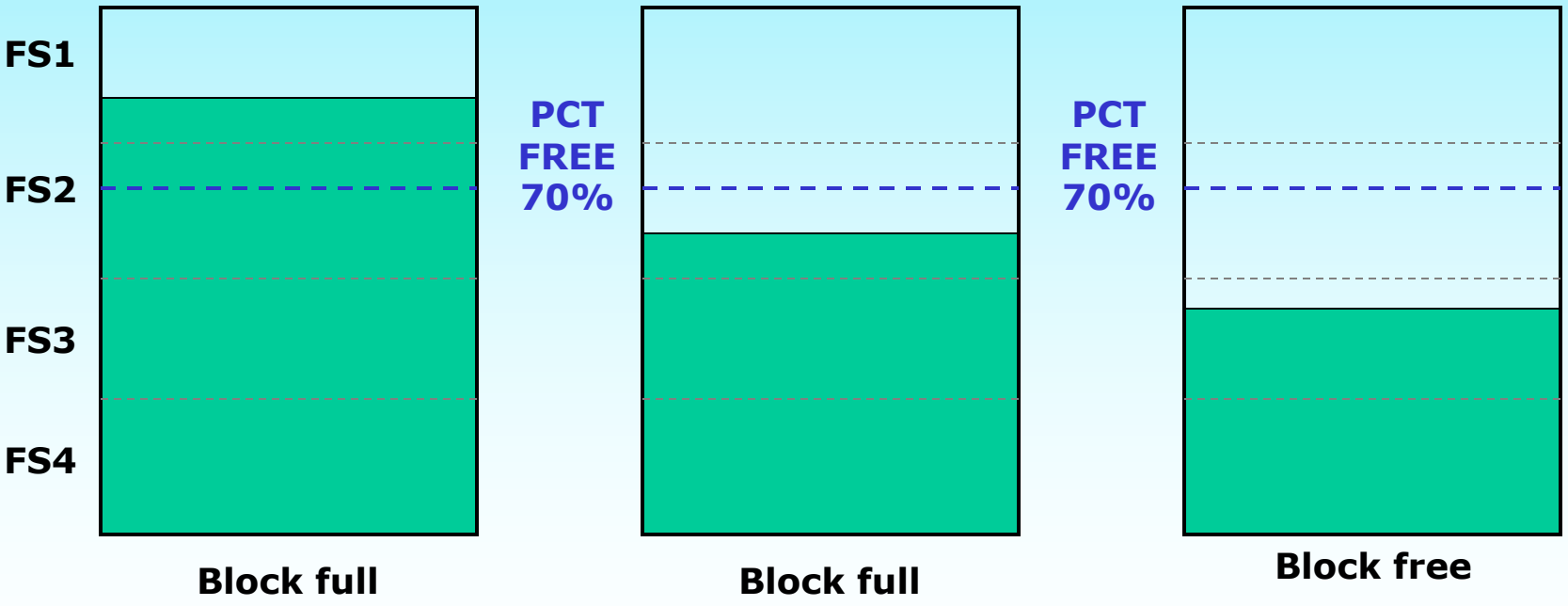
LHWM vs HHWM

- All blocks are formatted below LHWM
- All blocks are unformatted above HHWM
- Some blocks are formatted in between
- Unformatted blocks issue



Block State Transition

- Block space usage must drop to another freeness state (FS) below PCTFREE in order to get free



Searching for Space

1) Use L2 hint in seg. header to begin search
   - if not cached DBA
   - lock L2 BMB in shared mode
2) Find most free L1 BMB in L2 block
   - requests hashed by instance_number, PID
   - if no free enough L1 BMB, repeat with next L2
3) Build L1 array with enough free space
   - max 10 BMBs with correct instance affinity
   - if not enough space in L1, get another L2
   - L1 BMBs can be “stealed” from other instances
4) Extend the segment and release L2 lock

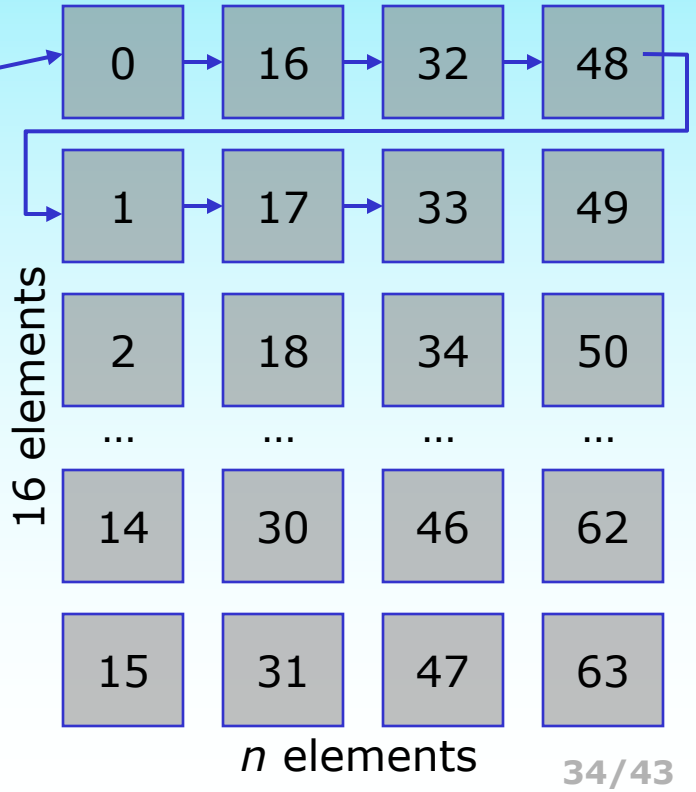
RAC: Stealing Blocks

1) If instance owning L1 BMB is dead, then steal the BMB
2) If instance is live, do a consistent read of the L1 BMB block
   - If sufficient time has passed since L1 BMB allocation or last "steal", steal the BMB
   - Controlled by *_inst_locking_period*, *_last_allocation_period* parameters
   - If BMB can't be stolen, skip to next
3) Bump up HWM

Searching for Space 2

- There is a Grid in Oracle9i as well!
- …but only meaning that a 2-dimensional array is made for searching free datablocks

1) Get L1 shared mode
   - hashed by PID
2) Scan array in steps
   - find 5 candiate blocks
   - skipping *n* elements in grid
3) Format unformatted blks
   - reget BMB in EXCL mode



Allocating Space in Datablocks

- Try to acquire a candidate block in NOWAIT mode
  - If a block is already pinned, skip it
  - Try NOWAIT on 5 blocks
  - if it fails, release L1 BMB lock and try to pin datablock normally with WAIT
- Unformatted blocks encountered during search are formatted and used
  - Reget L1 BMB in exclusive mode (FB enqueue)
- Setting HHWM and LHWM

PCTFREE Recalculation

- There is no automatic PCTFREE recalculation in case of ALTER TABLE in ASSM
- L1 “freeness” values are updated on subsequent DML access
- Manual segment level recalculation using dbms_repair.segment_fix_status
  - Recalculates statistics if with default parameters
  - Can change specific block freeness values manually

Converting to ASSM

- Very simple:

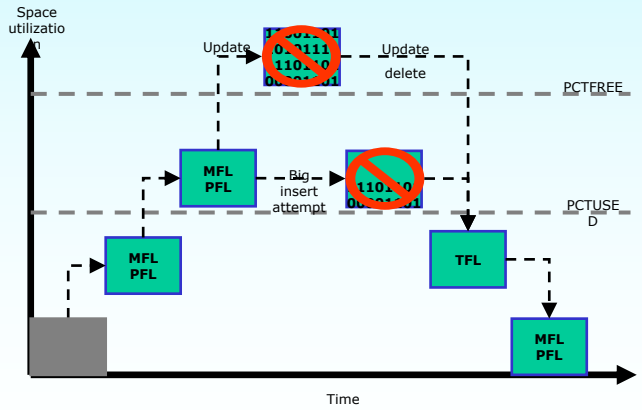
```
SQL> alter table t move tablespace users
nologging;

Table altered.
```

- COMPATIBLE parameter at least 9.0.1
- For ASSM LOB segments -> 9.2.0
- Possible segment growth
- Check for CLASS# 8, 9, 10 blocks from V$BH or X$BH

Freelist Pros & Cons

Pros:

- Virtually no space overhead
- Mature functionality
- Tunable



Cons:

- Contention on header blocks
- Default settings unreasonably low
- Premature unlink
- Large deletes - non-distributed list
- Unnecessary HWM bumping

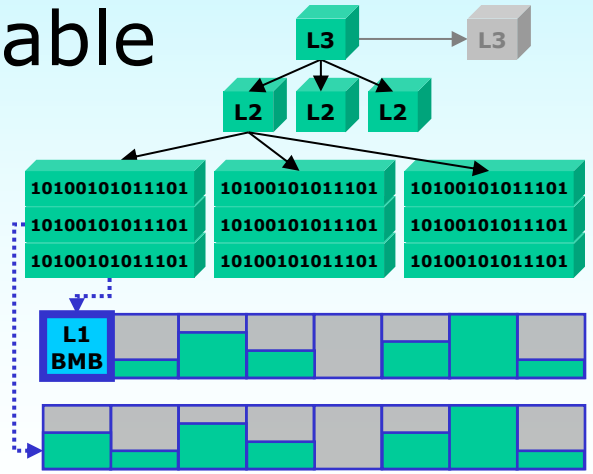
ASSM Pros & Cons

Pros:

- Reducing contention on freelist buffers & segment header
- Great for RAC
- Good for varying witdth rows
- Easy to set up
- Internal maintenance

Cons:

- Space usage
- Slower for FTS
- Fresh functionality
- Automatic - not tunable

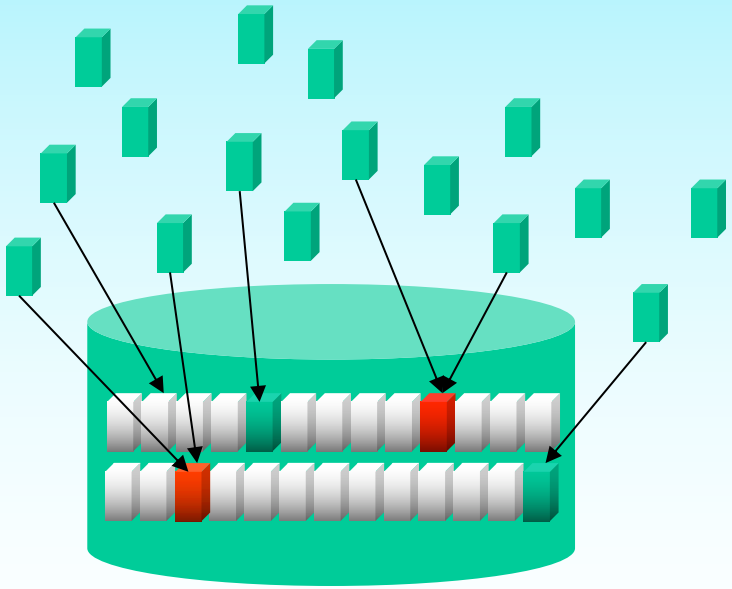


Freelists vs ASSM in Performance

- Contradicting results from users
- Everything depends on data usage patterns and workload
- Just creating and scanning a table doesn't give correct estimate of performance (de)improvements
  - Segment size is larger initially, but afterwards?
- In RAC it is not enough just to run concurrent insert on two instances
  - Adding and removing nodes, different workloads

Conclusion

- Indexes still have contention problem!
- If can afford being lazy, go with automatic
- If you want control, go with manual



Freelists vs ASSM in Oracle9i

Questions?

Freelists vs ASSM in Oracle9i

Tanel Poder

Thank you!

http://www.tanelpoder.com