Beyond the Happy Path

Dark Corners, Edge Cases, and Disaster Areas

SOUG Day 01/2025
2025-03-26
Mobiliar, Bern

Christoph Lutz
Swisscom

Twitter: @chris_skyflier
Bluesky: @christophlutz.bsky.social
Github: Christoph-Lutz

swisscom

What was the oldest Computer?

An apple with very limited memory …

… owned by Adam and Eve …

Just one byte and everything crashed!

Before we start …

This is a low-level technical presentation about internal and undocumented behavior

Beware that:

- Things can change across different versions and patch levels
- My observations, findings and interpretations may be inaccurate or wrong
- Some of the techniques shown in this presentation are dangerous – use them at your own risk!
- Examples and demos were tested in the following environments:

  Oracle 19.23, 19.24 (Exadata 24.1.2, OEL 8.10, kernel 5.4.17-2136.330.7.5, bpftrace 0.16)

  Oracle 23.7.0.25.01 (VirtualBox Image, OEL 8.10, kernel 5.15.0-3.60.5.1, bpftrace 0.16)

Note: all examples and demos are available on Github

Transactions – Consistency

Transactions – Simple Example: Update/Update Conflict

Time	Session 1	Session 2	Data
T1	update tc set n = n+1 where n > 0;		N - 1 1 -> 2 2 2 -> 3
T2		update tc set n=999 where n=2; (blocks)	N - 1 2 -> ? 2 3 -> ?
T3	commit;		What is the result after Session 1 commits at T3?

Answers

1) 1, 2 ?
2) 999, 3 ?
3) 2, 999 ?
4) 2, 3 ?
5) Error ?
6) I don't care !

Transactions – Update/Update Conflict

Isolation Level:
READ COMMITED

Oracle

```
  N
---
999
  3
```

Postgres

```
  n
---
  2
  3
```

Which one is correct?

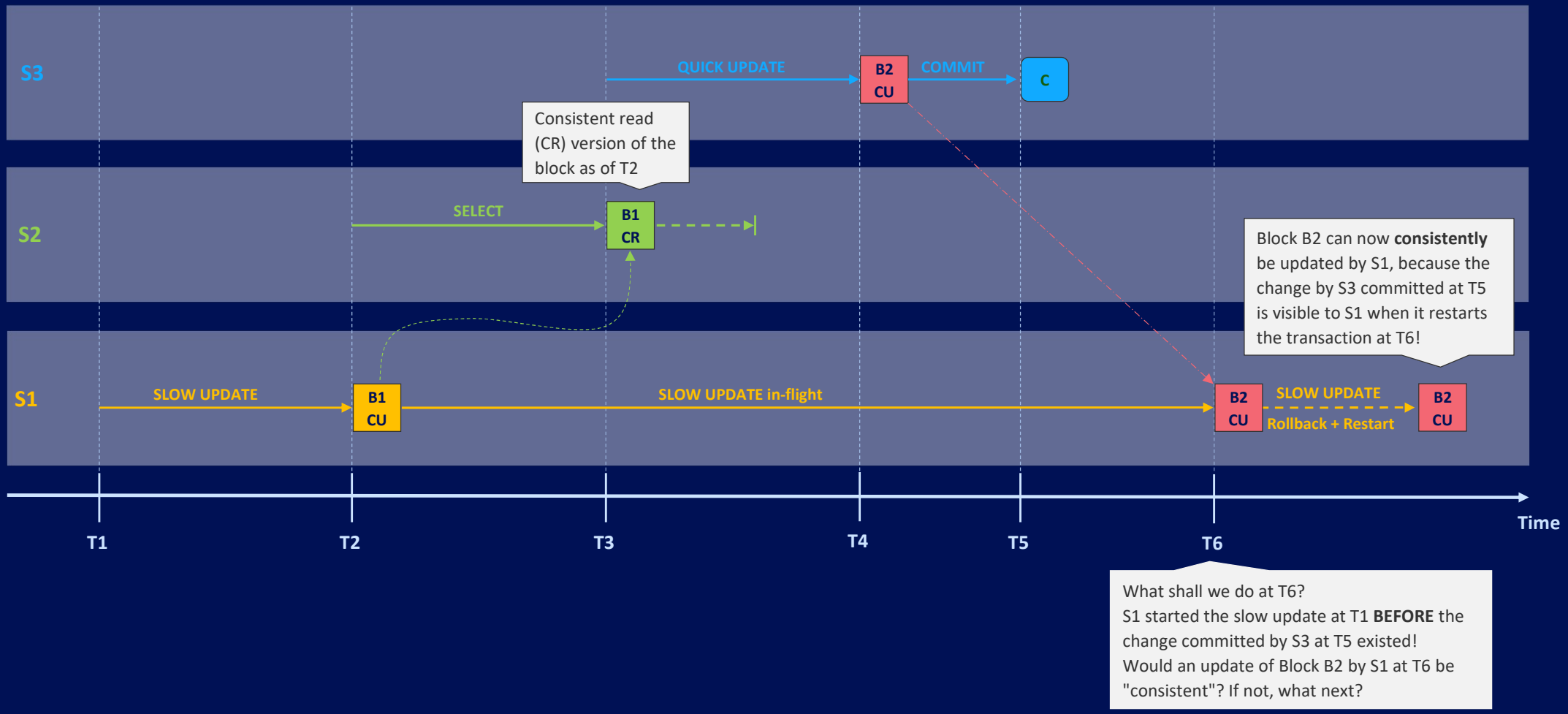
Postgres explanation: cf. Franck Pachot, READ COMMITTED anomalies in PostgreSQL , 2021-07-13

DEMO
tc01a, tc01b

Oracle Design Principle

Readers don't block writers,
writers don't block readers.

Write Consistency – Update/Update Conflicts



Write Consistency – Conflict Detection

CR == CUR ?

YES:

Modify row

NO:

Restart!

Write Consistency – Three Pass Algorithm

Updates are processed using a three pass algorithm, divided into three phases:

- **Phase 1: NOT LOCKED**
- **Phase 2: LOCK**
- **Phase 3: ALL LOCKED**

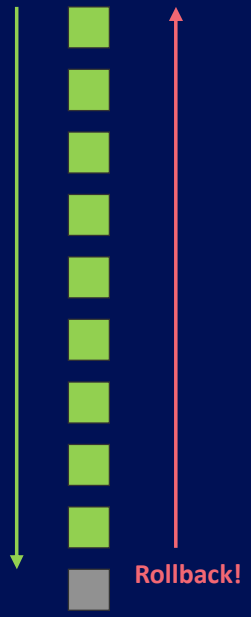
In every phase: CR result == CUR result ?

Restarts (or errors) occur when CR and CU return different results!

Three Pass Algorithm – Mechanics (Happy Path)

Phase 1: NOT LOCKED

"Optimistic Approach": Let's just try and attempt to **update all rows** …



Phase 2: LOCK

"Pessimistic approach": attempt to **lock all rows** (similar to SELECT FOR UPDATE)

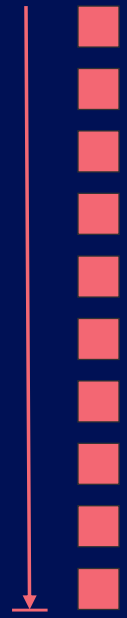
This phase can be **restarted** multiple times if not all the rows can be locked in one pass!

On every pass, use a new snapshot SCN.

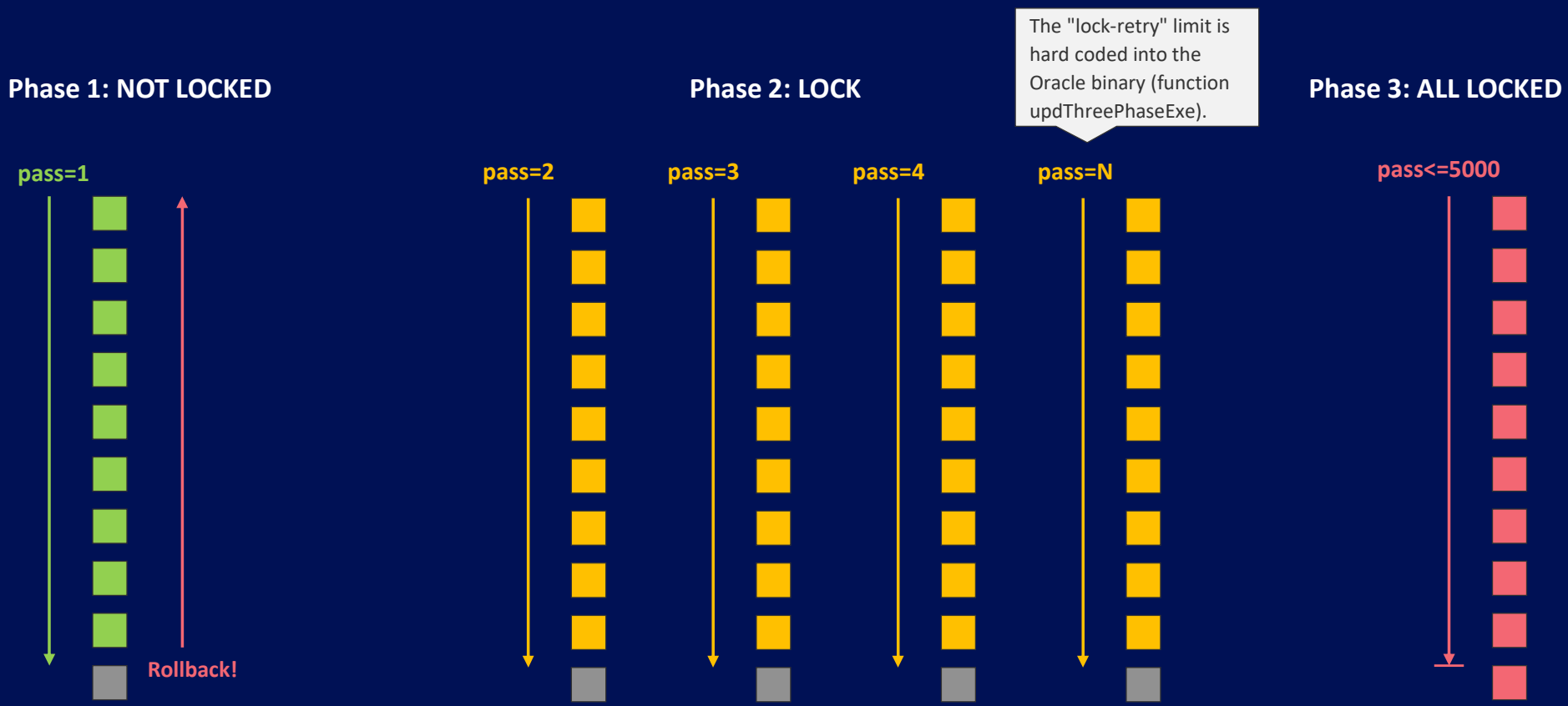
Phase 3: ALL LOCKED

Update all locked rows.

Use the snapshot SCN from the LOCK phase.

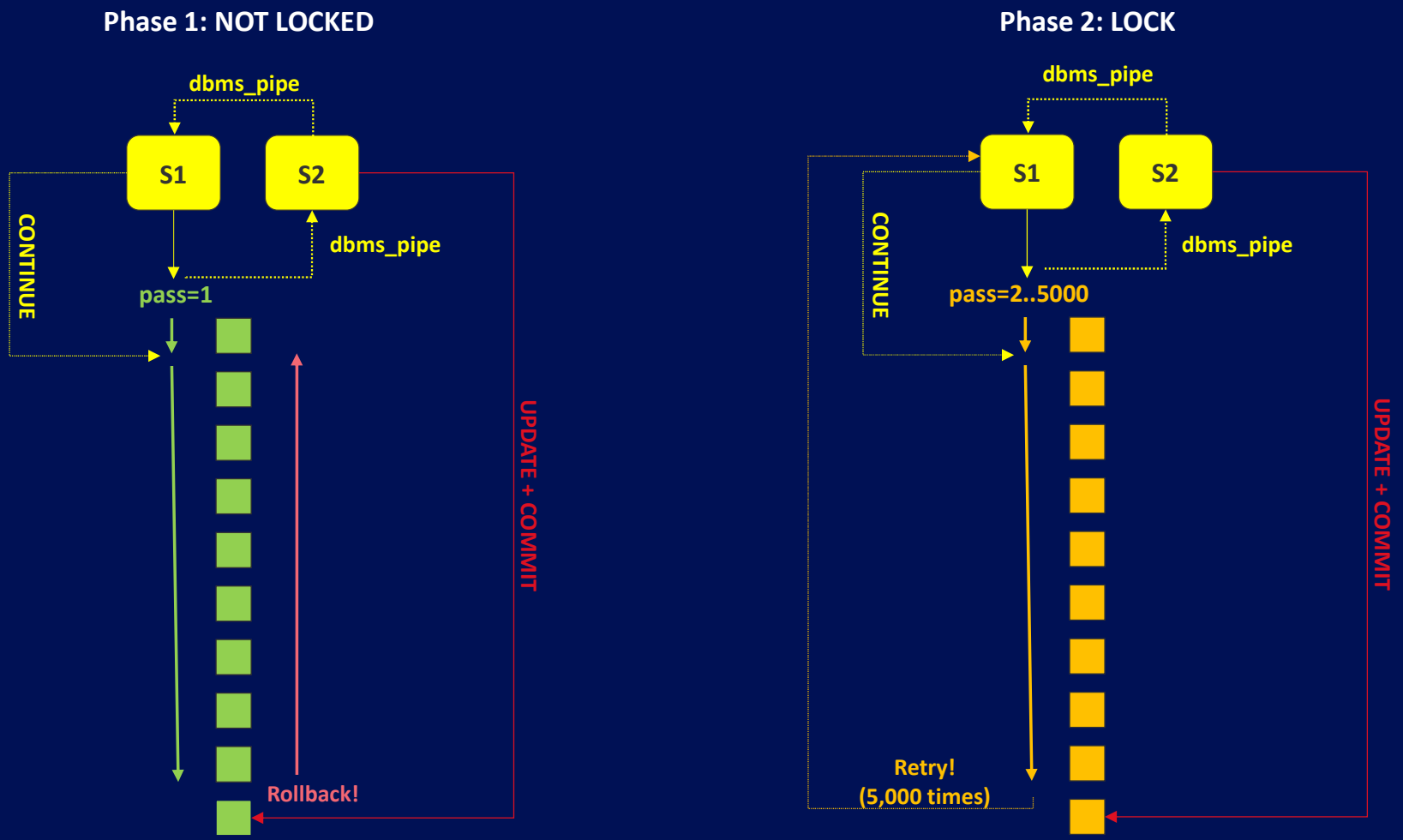


Three Pass Algorithm – Mechanics (Unhappy Path)



Oracle will fail with an ORA-600 [13013][5001] after 5,000 "lock-retry" attempts!

Lock-Retry Attempt Failure – Demo



Lock-Retry Attempt Failure – Demo

```
update tc_restart t set t.n = n+1 where test_func(t.n) > 0;
```

```
function test_func(p_n number) return number as
       l_msg varchar2(32);
begin

   if p_n = 1 then
          send(PIPE_SES2, CMD_UPDATE);
          receive(PIPE_SES1, l_msg);
   end if;

   return p_n;

end test_func;
```

DEMO
tc02

Update Restart Overhead – Session Statistics

TABLE_NAME	BLOCKS	NUM_ROWS
TC_RESTART	5	3

Restarts can come at high overhead!

transaction rollbacks..... 1

rollback changes - undo records applied..... 4

session logical reads..... 45,113

db block gets..... 15,030

db block gets from cache..... 15,030

db block gets from cache (fastpath)..... 15,027

consistent gets..... 30,083

buffer is pinned count 7

buffer is not pinned count 10,073

latch: cache buffers chains 166,842

calls to kcmgcs 5,013

calls to get snapshot scn: kcmgss 5,034

table scans (short tables) 5,008

table scan rows gotten 15,072

table scan blocks gotten 25,016

Write Consistency – Restart Observability

Refer to Appendix G for examples on enabling SQL trace or SQL Monitor at query level.

ASH

Write consistency restarts are not logged in ASH.

ASH columns sql_exec_id and sql_exec_start don't change on restart.

SQL Trace

With plan_stat enabled, row source statistics will log restarts (visible in the "str=<n>" field)

SQL Monitor[1]

SQL Monitor will log re-starts.

However, SQL Monitor only kicks in for PX or if a query consumes >5 sec of DB time!

[1]SQL Monitor query: Tanel Poder, TPT Github Repository, Script sqlmon_restarts.sql

Bpftrace Script update-restarts.bt

```
~# BPFTRACE_CACHE_USER_SYMBOLS=1 ./update-restarts.bt
Attaching 18 probes...

Tracing dml restarts... Hit ^C to stop.

TIME                RESTART_TIME        INSTANCE        PID     COMM            SID   USERNAME   SQL_HASH   OCT PLSQL_OBJ PLSQL_SUBID  DEP ATTEMPTS L-RETRIES STATUS   ERR1   ERR2          CR        CU     ELA_MS
2025-01-16 13:43:39 2025-01-16 13:43:34 TCTEST19        35687   oracle_35687_tc 1410  TEST       1911506387   6    469860           2    1     5002       5000 ORA-600 13013  5001       30269     15014      4836
2025-01-16 14:00:31 2025-01-16 14:00:31 TCTEST19        215116  oracle_215116_t 287   TEST       2109685233   6         0           0    0        3          1 ORA-600 13030  3             70        53        10
2025-01-16 14:01:55 2025-01-16 14:01:55 TCTEST19        215116  oracle_215116_t 287   TEST       2109685233   6         0           0    0        3          1 SUCCESS n/a    n/a           32        72         9
```

TIME:
Current date/time when a restarted update statement completed or failed.

RESTART_TIME:
Time when an update statement got restarted.

INSTANCE:
Oracle sid / instance name.

PID:
Oracle os pid.

COMM:
Oracle process command name.

SID:
Oracle session id.

USERNAME:
Oracle db user name.

SQL_HASH:
Sql hash value (v$sql.hash_value).

OCT:
Oracle command type (v$session.command).

PLSQL_OBJ:
Plsql object id (v$session.plsql_cobject_id);
maps to dba_objects.object_id.

PLSQL_SUBID:
Plsql subprogram id (v$session.plsql_subprogram_id);
maps to dba_procedures.subprogram_id.

DEP:
Dependency level (dep=n in sql trace) at which the restarted update got executed.

ATTEMPTS:
This counts the number of calls to updaul ('update - attempt update or lock all rows').

L-RETRIES:
This counts the number of lock-retries (attempts to lock all rows in the LOCK phase).

STATUS:
Status of the restarted update statement; this is either SUCCESS (restart ok) or ORA-600 (restart failed).

ERR1:
In case of ORA-600, the first error argument.

ERR2:
In case of ORA-600, the second error argument.

CR:
number of consistent reads performed by a restarted update statement .

CU:
Number of current gets performed by a restarted update statement

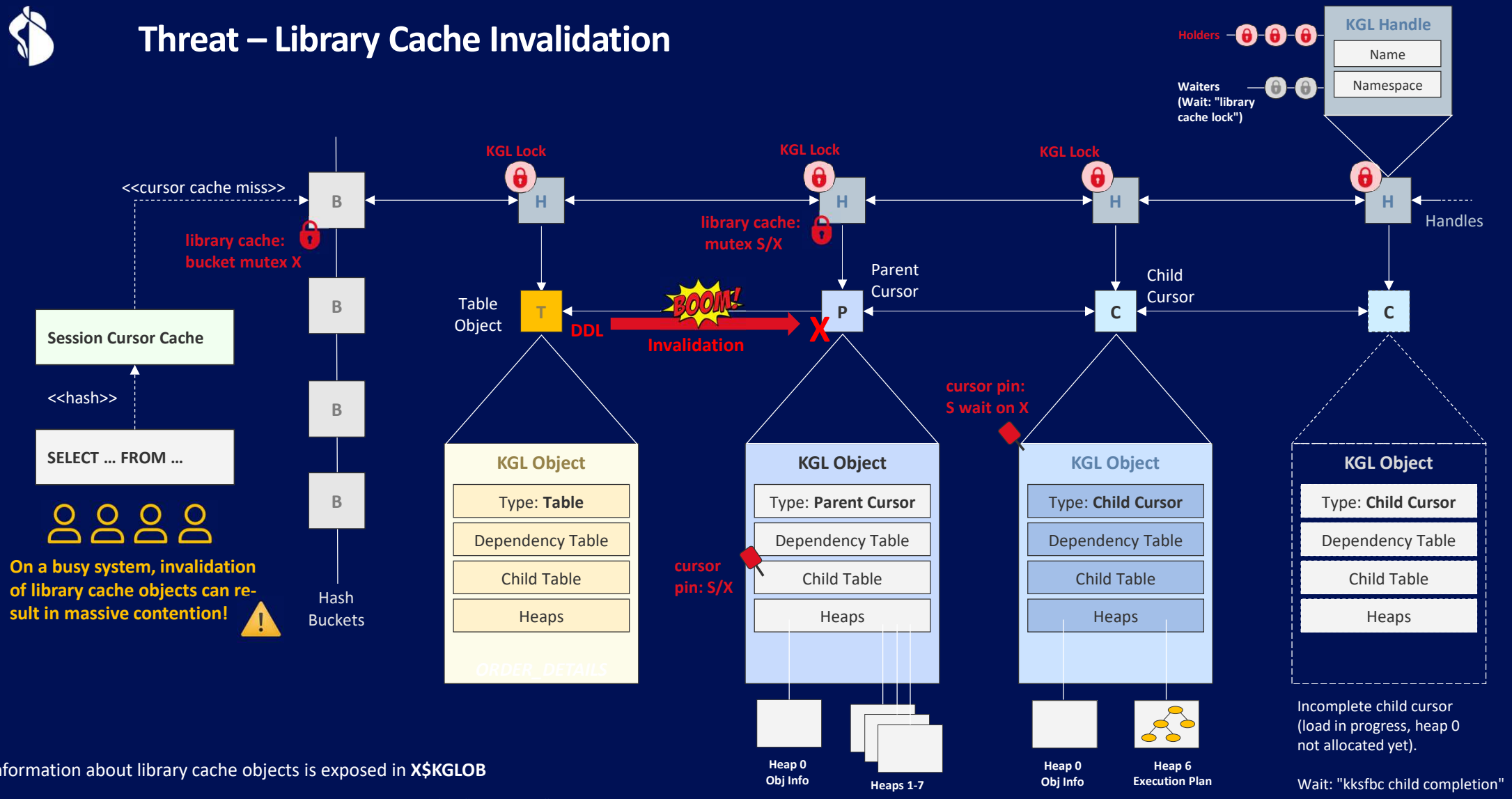
ELA_MS:
Elapsed time of a restarted update statement (in ms).

Script source: Christoph Lutz, Github Repository, Script update_restarts.bt

Library Cache Invalidation

Library Cache Invalidation

Threat – Library Cache Invalidation



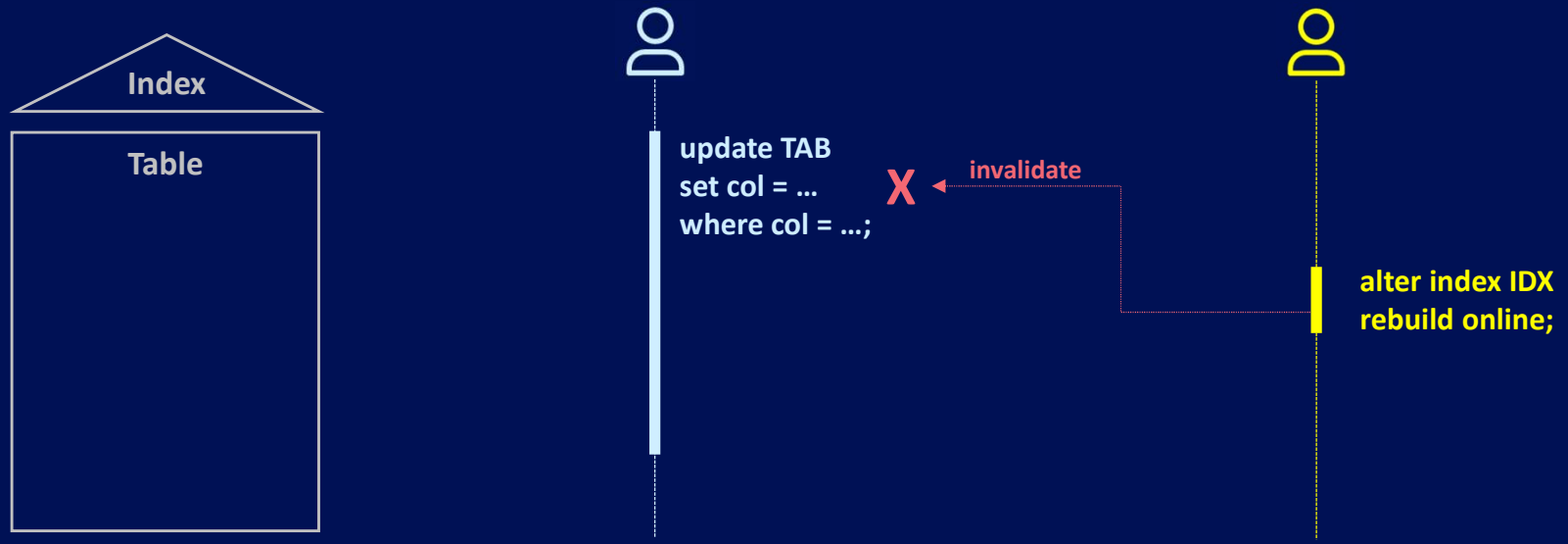
Information about library cache objects is exposed in **X$KGLOB**

Note: Library cache structures are very complex, this shows a simplified "conceptual" view! Hoogland 2017 & Hoogland 2020 are excellent resources on this topic.

Is an Index Rebuild an ONLINE Operation?

1. YES!
2. NO!
3. It depends …

Example: Cursor Invalidation – Online Index Rebuild: Non-Partitioned Table

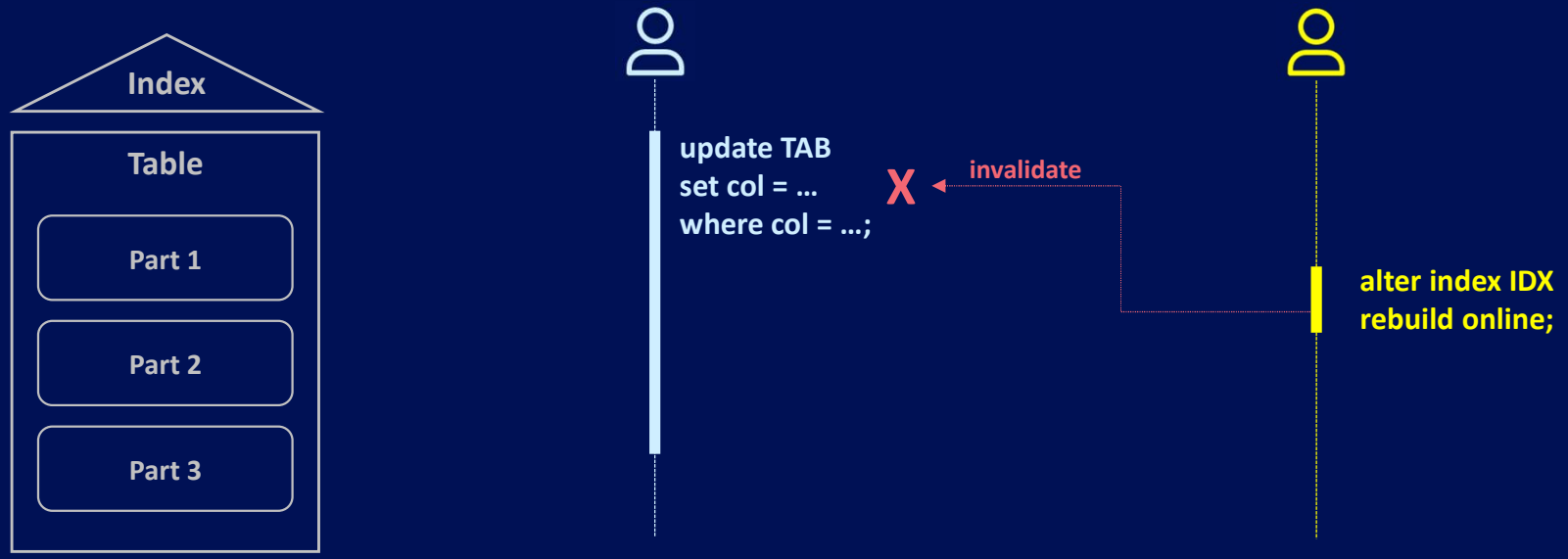


Q: What will happen when the UPDATE cursor gets invalidated by the INDEX REBUILD?

A: Oracle will re-parse the query at runtime (this shows as "mis=1" in SQL trace).

DEMO
tc03a

Example: Cursor Invalidation – Online Index Rebuild: Partitioned Table



Q: What will happen when the UPDATE cursor gets invalidated by the INDEX REBUILD?

A: Oracle will raise an ORA-14403. This is an internal error, not visible to users, BUT …

Threat – ORA-14403

```
14403, 00000, "cursor invalidation detected after getting DML partition lock"

// *Document: NO

// *Cause:  cursor invalidation was detected after acquiring a partition lock
//          during an INSERT, UPDATE, DELETE statement, or cursor invalidation
//          was detected after refresh[illegible]ead snapshot for
//          SELECT FOR UPDATE or SELEC[illegible]e returned to the client.
//          This error is never returned to user, because is caught in
//          opiexe() and the statement is retried.

// *Action: nothing to be done, error should never be returned to user
```

This mechanism exists to guarantee [...] correctness (MOS Doc ID 2057107.1).

Cursor invalidation can trigger DML statement retries!

DEMO
tc03b

s. also: My Oracle Support, Complex Insert Intermittently Slower on Partitioned Table as a Result of Cursor Invalidation (Doc ID 2057107.1), 2022-08-29

ORA-14403 – Restart Observability: V$SYSSTAT & V$SESSTAT

DML statements retried

When a long-running DML is executing, the cursor may get invalidated due to some concurrent DDL on one of the cursor's dependencies. In this case, an internal ORA-14403 error is thrown and is caught and cleared in one of the calling functions.

The current work is rolled back and the DML is restarted without the user being notified of this.

The statistic counts the number of times that the thrown, caught, and cleared (ORA-14403) sequence occurred for DML statements. Should a DML vary widely in execution time, check this statistic to see if it increments during the DML execution. If so, then concurrent DDL may be the cause of the extra elapsed time.

- **The statistic counter is incremented in opiexe.**
- **The statistic counter is not incremented for write consistency restarts (handled in updThreePhaseExe)!**

Source: Oracle 23c, Database Reference, Section E.2: Statistics Description

Write Consistency Restarts vs DML Retries – Code Paths (UPDATE)

Write consistency restarts and DML retries can both occur during a single statement execution (s. also Appendix A). Anyway, this is something for another day … :-)

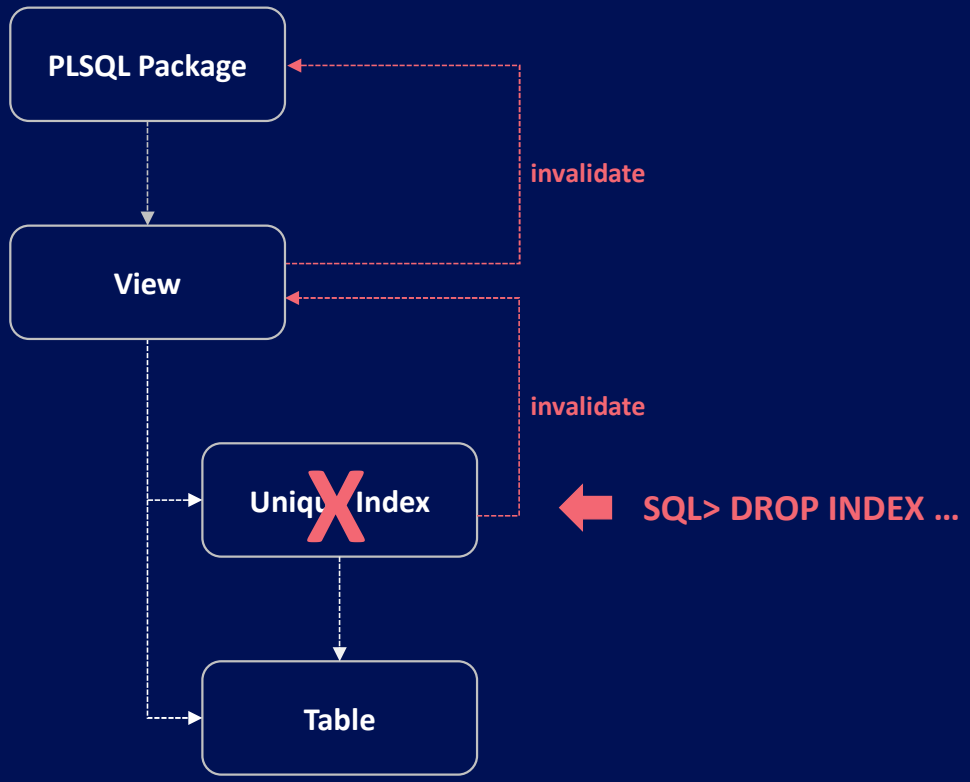
Write Consistency Restart

```
-> opiexe
   -> updexe
      -> updThreePhaseExe
         -> updaul
            Phase 1: NOT LOCKED
         <- updaul
         -> updaul
            Phase2: LOCK
         <- updaul
         -> updaul
            Phase 3: ALL LOCKED
         <- updaul
      <- updThreePhaseExe
   <- updexe
<- opiexe
```

DML Retry

```
-> opiexe
   -> updexe
      -> updThreePhaseExe
      <- updThreePhaseExe
   <- updexe
   -> updexe
      -> updThreePhaseExe
      <- updThreePhaseExe
   <- updexe
   -> updexe
      -> updThreePhaseExe
      <- updThreePhaseExe
   <- updexe
<- opiexe
```

Example: PLSQL Package Invalidation – What could possibly go wrong?



Example: PLSQL Package Invalidation – What could possibly go wrong?

```
create or replace package pkg_shared as

    function helper(p_n number) return number;

    procedure dequeue(p_payload in out varchar2);

end pkg_shared;
/

…

loop
  dbms_aq.dequeue(
    queue_name          =>'&&queue_name',
    dequeue_options     => l_dequeue_options,
    message_properties  => l_message_properties,
    payload             => l_message,
    msgid               => l_message_handle);
  …
 end loop
 …
```

DEMO

tc04

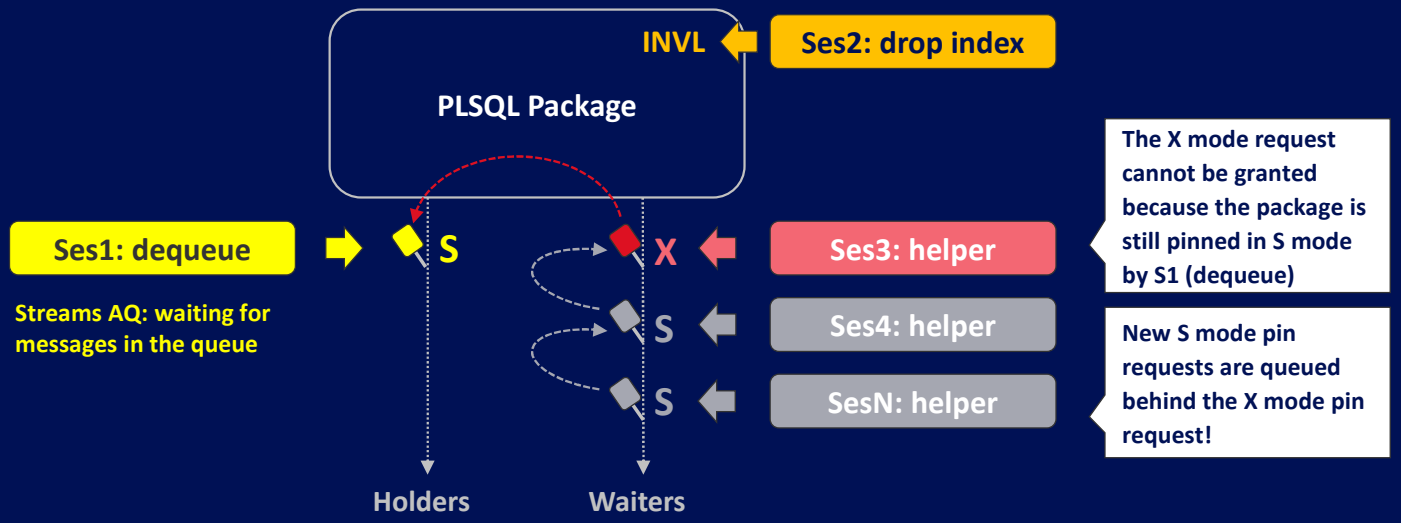
Example: PLSQL Package Invalidation – "library cache pin" Contention

```
SQL> @ashtop.sql event2,substr(to_char(p3,'xxxxxxxxxxxxxxx'),-1),blocking_session
     "event='library cache pin'" "systimestamp-(2/1440)" "systimestamp"

  Total
  Seconds     AAS %This   EVENT                     SUBS BLOCKING_SESSION
 -------- ------- ------- ------------------------- ---- ----------------
     6000    50.0   98% | library cache pin           2            (???)
      121     1.0    2% | library cache pin           3             1430



SQL> @ash_wait_chains.sql event2||'('||substr(to_char(p3,'xxxxxxxxxxxxxxxx'),-1)||')'||':'||blocking_session
     1=1 "systimestamp-(2/1440)" "systimestamp"

%This     SECONDS     AAS WAIT_CHAIN
------ ---------- ------- --------------------------------------------------------------------------------
97%         6000    50.0 -> library cache pin (2):        (???)
 2%          120     1.0 -> library cache pin (3):1430 -> [idle blocker 1,1430,12443 (sqlplus@...)
```

- **"library cache pin" S mode blocking session(s) not visible in ASH!**
- **"Streams AQ: waiting for messages in the queue" is an idle wait event, so not logged in ASH!**

Scripts source: Tanel Poder, TPT Github Repository, Script ashtop.sql and Tanel Poder, TPT Github Repository, Script ash_wait_chains.sql

Example: PLSQL Package Invalidation – "library cache pin" Contention




```
SQL> oradebug dump library_cache_object 16 0x213E4F378

LibraryHandle:
 Address=0x213e4f378 Hash=14c2eb8a LockMode=X PinMode=S
  LoadLockMode=0 Status=INVL Subpool=1

 ObjectName:  Name=TPDB002.TEST.PKG_SHARED
  …
  Block:  #='4' name=PLMCD^14c2eb8a pins=1 Change=NONE
```

```
SQL> select kgllkuse, kgllkmod, kgllkreq, kgllktype
     from dba_kgllock where kgllkhdl like '%213E4F378%';

KGLLKUSE           KGLLKMOD   KGLLKREQ KGLLKTYPE
---------------- ---------- ---------- ---------
000000042D6D2518          1          0 Lock
000000042D6D2518          2          0 Pin
000000042D6D9CD0          0          3 Pin
000000042D6D9CD0          3          0 Lock
…
```

Deferred / Rolling Cursor Invalidation

Rolling Cursor Invalidation

DDL

System Level
CURSOR_INVALIDATION=DEFERRED

Session Level:
CURSOR_INVALIDATION=DEFERRED

Statement Level:
ALTER INDEX … DEFERRED INVALIDATION

Optimizer Statistics

DBMS_STATS:
NO_INVALIDATE=>DBMS_STATS.AUTO_INVALIDATE

Refer to Appendix D for further details on rolling cursor invalidation with dbms_stats.

Many factors […] determine whether Oracle Database uses deferred invalidation.[1]

[1]Source: Oracle 19c, SQL Tuning Guide, Section 20.1.4.2: Cursors Marked Rolling Invalid

Does Deferred/Rolling Invalidation Prevent ORA-14403 Restarts?

1. YES!
2. NO!
3. It depends …

Does Deferred/Rolling Invalidation Prevent ORA-14403 Restarts?

Non-Partitioned Table

Table Stats Collection with NO_INVALIDATE=TRUE ..... ✓
Table Stats Collection with NO_INVALIDATE=FALSE ..... ✓
Table Stats Collection with AUTO_INVALIDATE ..... ✓

Online Index Rebuild X
Online Index Rebuild with DEFERRED INVALIDATON X

Partitioned Table

Table Stats Collection with NO_INVALIDATE=TRUE ..... ✓
Table Stats Collection with NO_INVALIDATE=FALSE ..... ✓
Table Stats Collection with AUTO_INVALIDATE ..... ✓

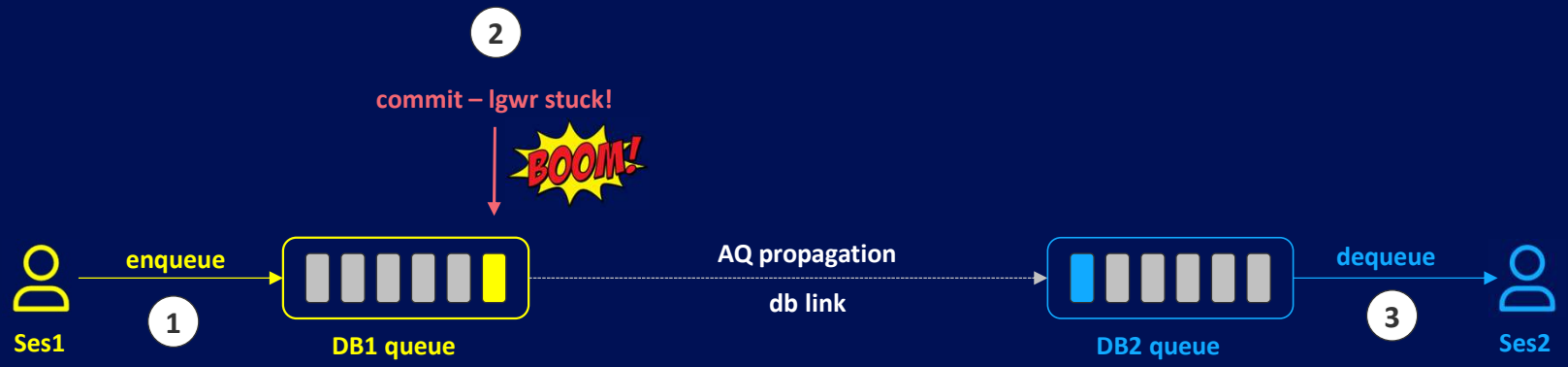
Part. Stats Collection with NO_INVALIDATE=TRUE ..... ✓
Part. Stats Collection with NO_INVALIDATE=FALSE ..... ✓
Part. Stats Collection with AUTO_INVALIDATE ..... ✓
Part. Truncate ..... ✓

Online Global Index Rebuild X
Online Global Index Rebuild with DEFERRED INVALIDATON X
Online Local Index Rebuild X
Online Local Index Rebuild with DEFFERRED INVALIDATION X

DEMO
tc05a, tc05b

Transactions – ACID

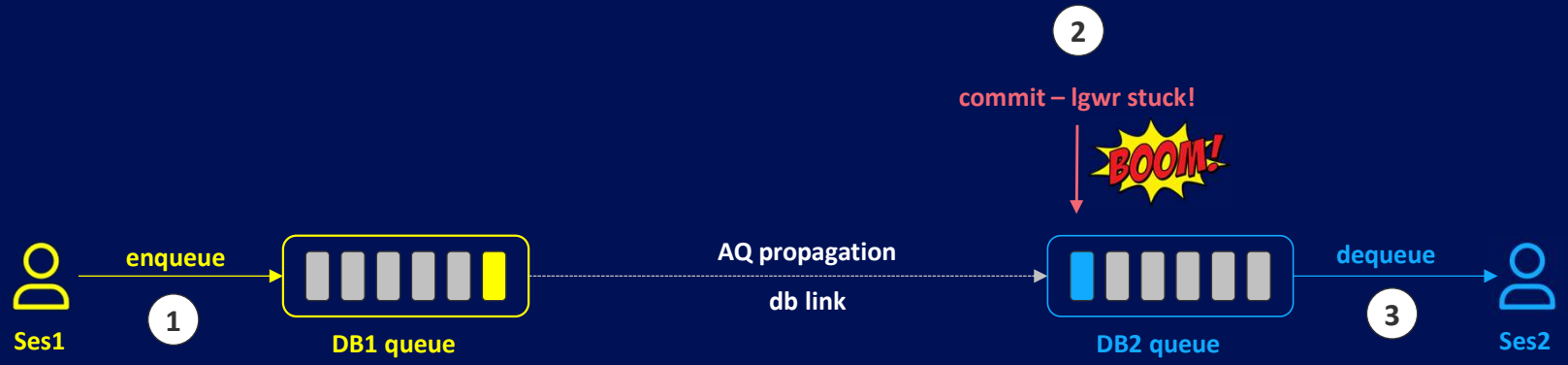
Example: AQ Propagation – Distributed System (1/2)



Will the message propagate if lgwr gets stuck when session1 on DB1 commits?

DEMO
tc06

Example: AQ Propagation – Distributed System (2/2)



Will the commit by session1 on DB1 return when lgwr on DB2 gets stuck?

DEMO
tc06

Transactions – Visibility and Durability

Time	Session 1	Session 2	LGWR
T1	update tc set n = n+1 ; commit;		
T2	(commit in-flight)		Start writing … Write gets stuck!
T3		select * from tc;	

Q1:
When exactly does a committed change become visible?

Q2:
Will session 2 see the committed or the uncommited change?

Q3:
What if the instance crashes between T1 and T3

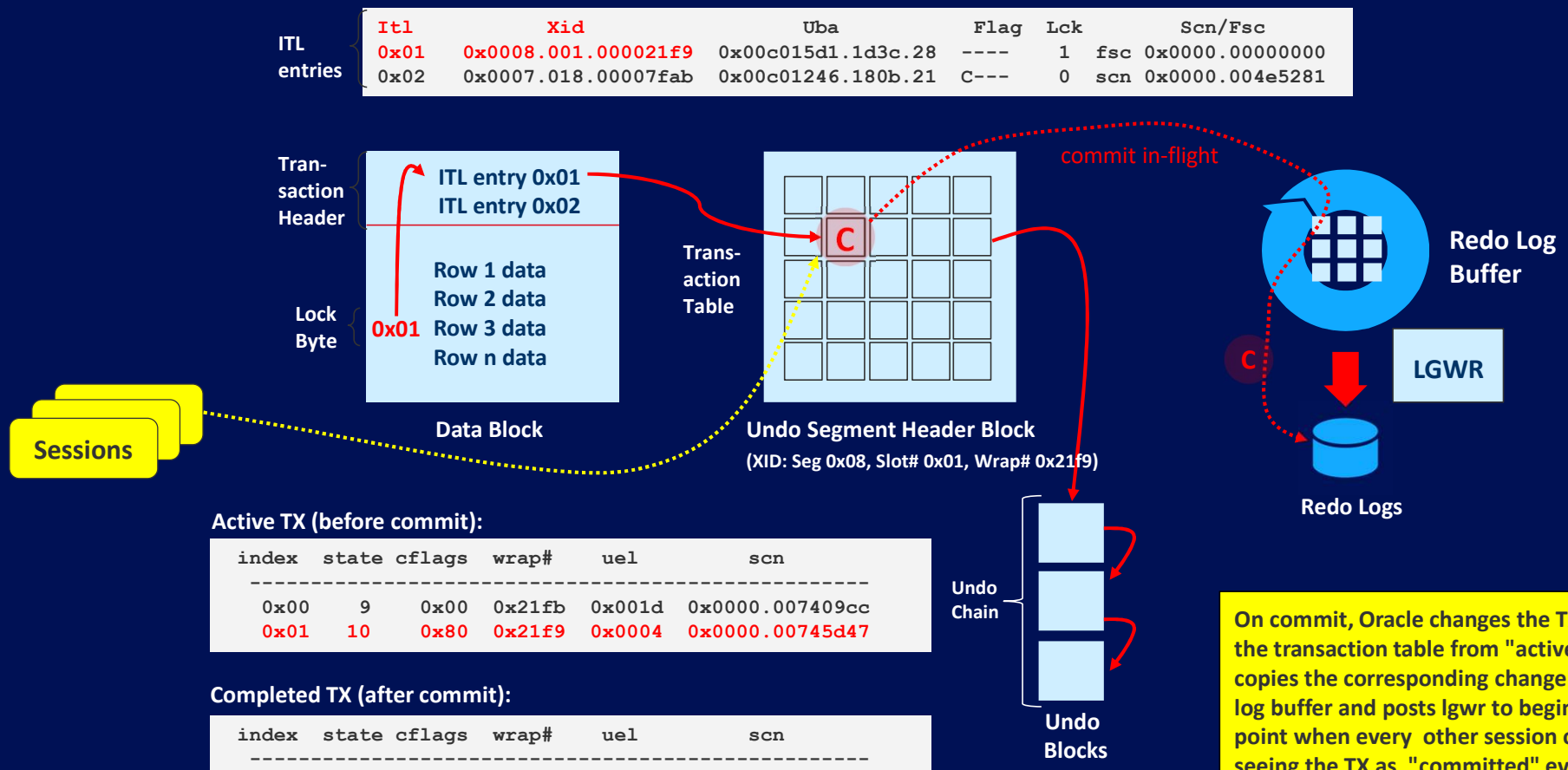
Test case source: Jonathan Lewis, Redo, 2011-08-19

Is the Oracle Transaction Behavior ACID Compliant?

A	C	I	D
Atomicity	Consistency	Isolation	Durability




Oracle Transaction Mechanics



Active TX (before commit):

index	state	cflags	wrap#	uel	scn
0x00	9	0x00	0x21fb	0x001d	0x0000.007409cc
0x01	10	0x80	0x21f9	0x0004	0x0000.00745d47

Completed TX (after commit):

index	state	cflags	wrap#	uel	scn
0x00	9	0x00	0x21fb	0x001d	0x0000.007409cc
0x01	9	0x00	0x21f9	0xffff	0x0000.007461ef

On commit, Oracle changes the TX state and flags in the transaction table from "active" to "committed", copies the corresponding change record into the redo log buffer and posts lgwr to begin writing. It is at this point when every other session on the system starts seeing the TX as "committed" even though the lgwr write I/O is still in flight!

Paranoid Concurrency Mode (23ai) – Undocumented

Refer to Appendix E for additional technical details

Parameter (new in 23ai)

paranoid_concurrency_mode

Default Value

FALSE

Description

Enable **strictly durable** query data gets

Documentation

None (yet)

Wait Event (new in 23ai)

log file sync - queried data

Parameter 1

block#

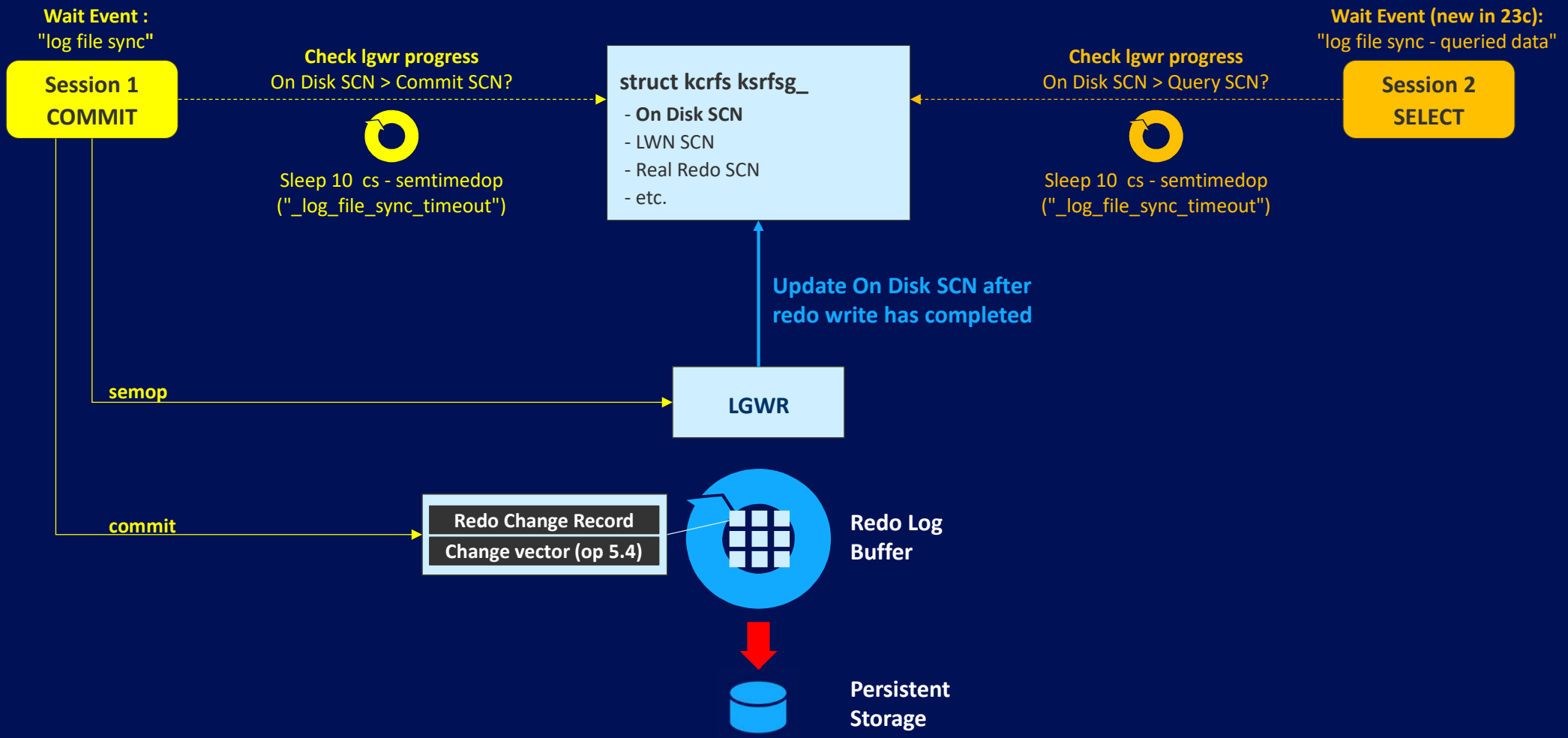
Parameter 2

thread scn

Parameter 3

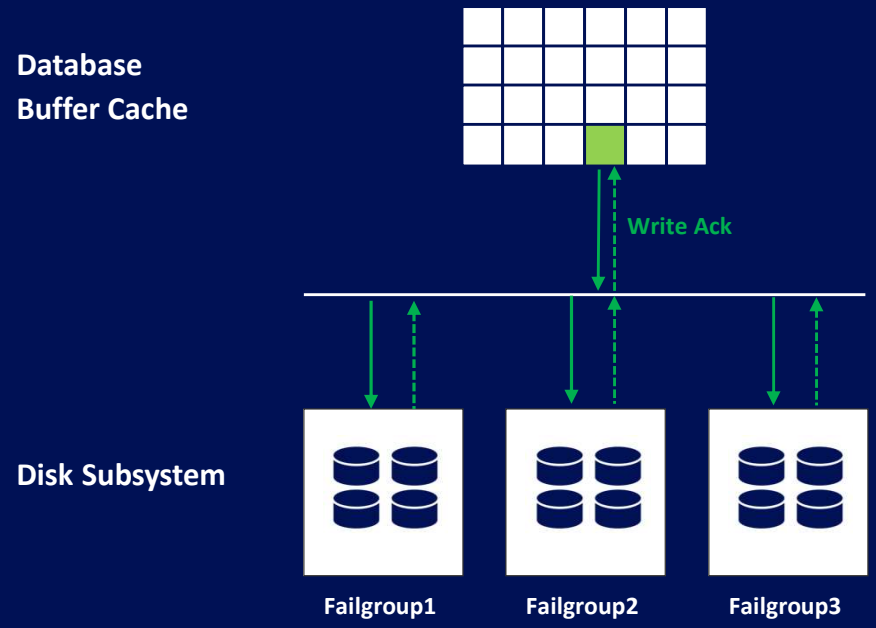
sync scn

Paranoid Concurrency Mode – Overview



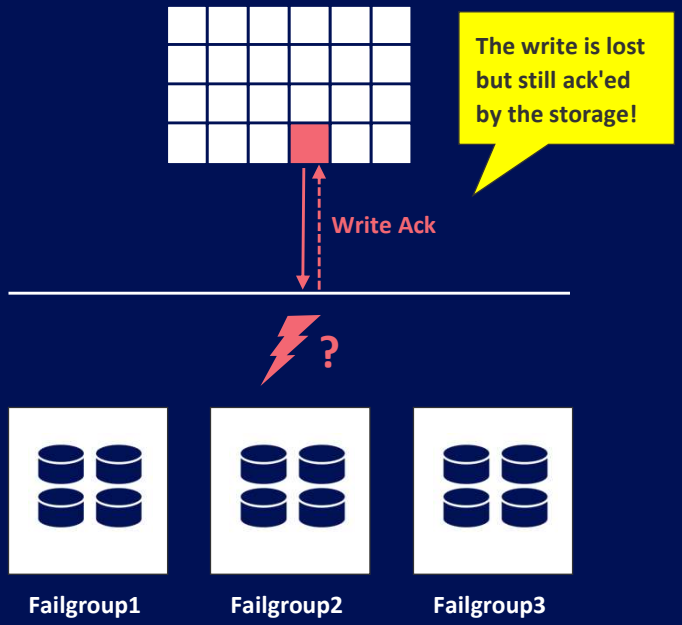
ASM Lost Writes

Normal Case – Happy Path

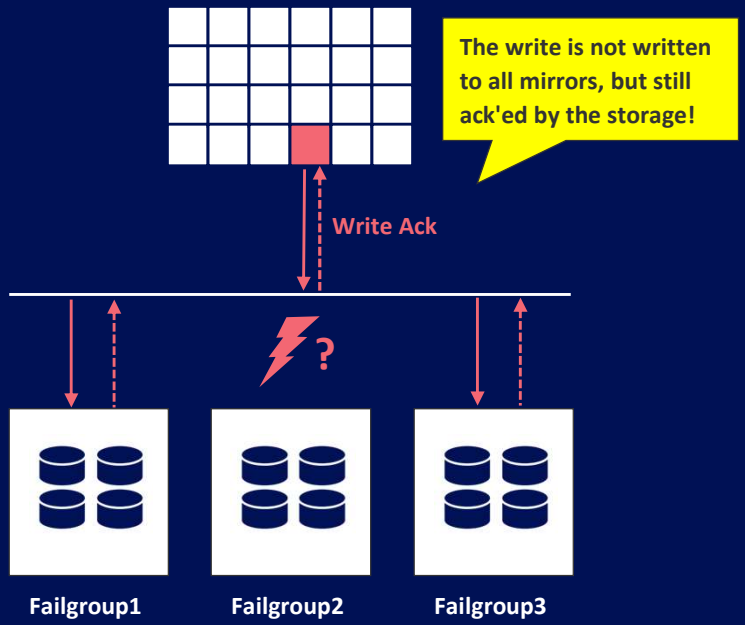


Error Case – Lost Write Scenarios

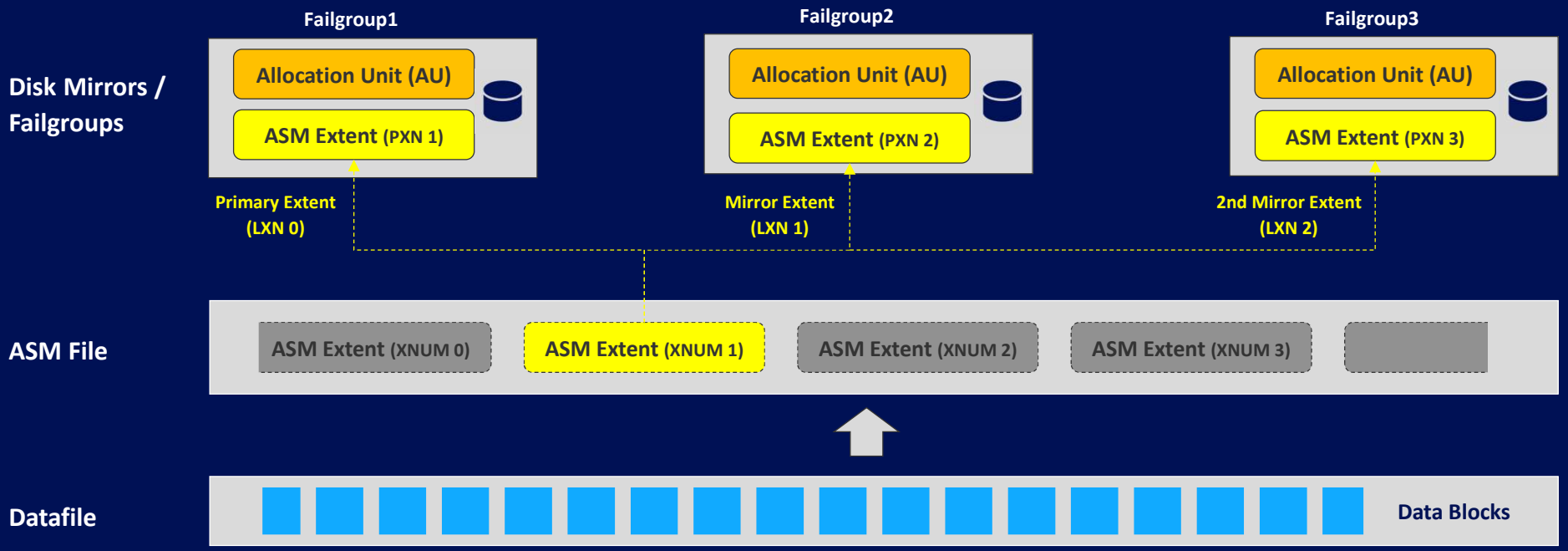
Lost Write



Lost Mirror Write



Partially Lost Writes – ASM Basic Structures



- ASM divides a datafile into multiple ASM extents
- ASM extents are mapped to allocation Units (AU) and mirrored across different disks / failgroups

- **PXN:** Progressive Extent Nr
- **XNUM:** (Virtual) Extent Nr
- **LXN:** Logical Extent Nr

Partially Lost Writes – ASM Mirroring

Basic Mechanism

ASM mirrors at the extent level.

Write operations write a copy of an extent to **all mirrors** (fail groups).

Read operations access only the **primary extent of a mirror**
(unless there is an IO error or preferred mirror reads have been configured).

Worst case:
RMAN backup read operations also only access the primary mirror and if that is damaged in a way not detected by ASM, your backups may become useless!

Questions

- What if a mirror incurs a lost write?
- What if multiple mirrors incur a lost write?
- What if all mirrors incur a lost write?

Source: Luca Canali, ASM Metadata and Internals, May 2014

ASM Metadata – X$KFFXP (Kernel File eXtent Pointers) Query

```
define asm_file_name='users.296.1163417749'

select
    xp.group_kffxp,    -- disk group number
    xp.disk_kffxp,     -- disk number
    xp.number_kffxp,   -- asm file number
    xp.pxn_kffxp,      -- physical extent number
    xp.xnum_kffxp,     -- virtual extent number
    xp.lxn_kffxp,      -- logical mirror extern number (0=primary, 1=secondary, 2=secondary)
    xp.au_kffxp,       -- allocation unit number
    to_number(at.value) au_size,
    to_number(at.value/1024/1024) au_size_mb
from
    v$asm_alias al,
    x$kffxp xp,
    v$asm_attribute at
where
    xp.number_kffxp = al.file_number
and upper(al.name) = upper('&&asm_file_name')
and xp.group_kffxp = al.group_number
and xp.number_kffxp = al.file_number
and at.group_number = xp.group_kffxp
and at.name = 'au_size'
order by
    xp.xnum_kffxp,
    xp.lxn_kffxp
/
```

This query will show the ASM extent to Allocation Unit (AU) mapping. It must be executed on the ASM instance!

Query modified from source: Norbert Debes, Secrets of the Oracle Database, p. 123, Apress, 2009

ASM Metadata – asmcmd mapblk

Map DB Block to ASM Extent:

floor(block# * db_block_size / AU_SIZE) = floor(783 * 8192 / 4194304) = **1**

Map DB Block to ASM Disk Offset:

((AU# * AU_SIZE) + (block# * db_block_size) - (XNUM * AU_SIZE)) / db_block_size =
((6693 * 4194304 + (783 * 8192) - (1 * 4194304)) / 8192 = **3427087**

```
# Example: File#/block# = 37/783
$ asmcmd mapblk +DATA1234/MYDB_123/F9DA9F2E43C0A555E053C44313C66087/DATAFILE/users.296.1163417749 783

Logic_Ext  Block_Size   Offset  Disk_Num  Path
        0        8192  3427087         8  o/100.106.100.40;100.106.100.41/DATA1234_CD_02_exa99cel02_m
        1        8192  3424015         4  o/100.106.100.38;100.106.100.39/DATA1234_CD_02_exa99cel01_m
        2        8192  3419407        16  o/100.106.100.42;100.106.100.43/DATA1234_CD_07_exa99cel03_m
```

Formula source: Norbert Debes, Secrets of the Oracle Database, p. 124, Apress, 2009

ASM Metadata – Exadata Complications

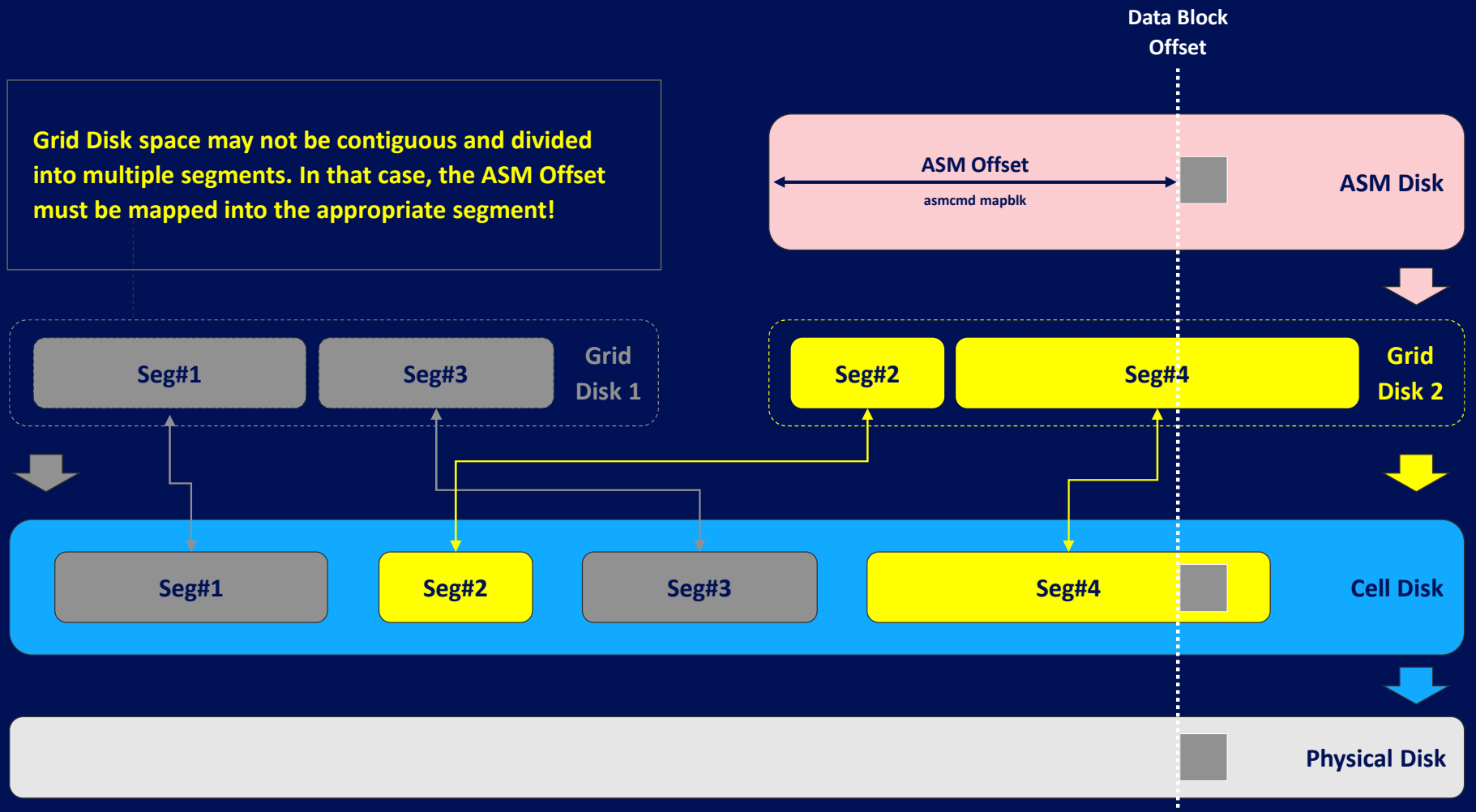
Exadata Disk Layers

- Multiple disk layers:
  ASM Disk -> Grid Disk -> Cell Disk -> Physical Disk
- Grid Disks start at a predefined offset on a Cell Disk
  (usually 32 MB for DATA disks on bare metal deployments)

Segmented Grid Disks

- Space on Grid Disks is not always contiguous
- Grid Disks can be composed of multiple segments (of varying sizes and at different offsets).
- Low-level Cell Disk details can be displayed with the undocumented cellutil utility.

Exadata Complications – Disk Structures

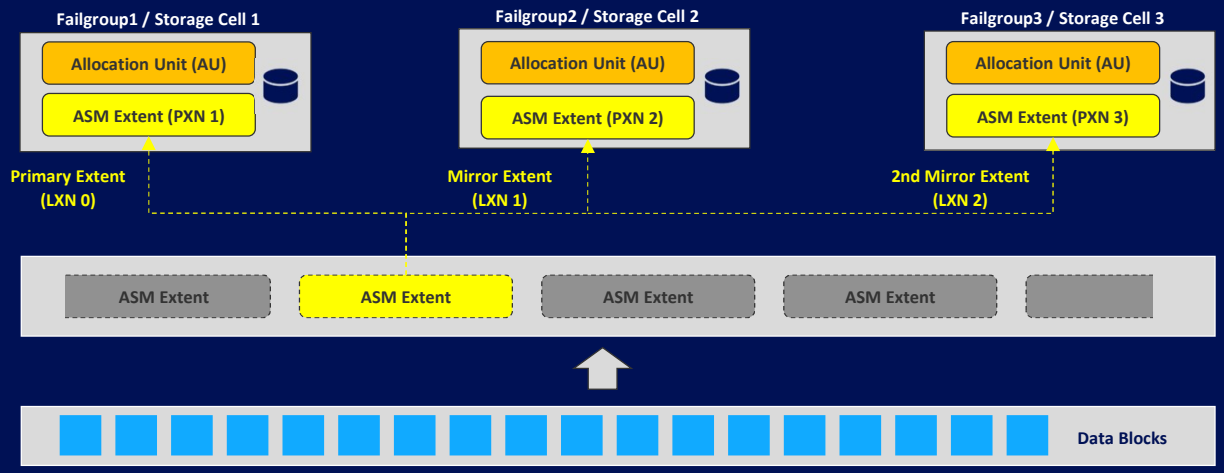


Exadata Complications – PMEM/XRMEM & Flash Cache

```
ALTER IORMPLAN dbplan=(
   (name=DB_UNIQUE_NAME,
         flashcache=off, flashlog=off,
         pmemcache=off,  pmemlog=off
   )
 )
```

Switch off PMEM and FLASH cache for testing!

ASM Scrubbing – Test Case



Extract current version of the block

```
dd if=/dev/sdc of=/tmp/783-old.dmp bs=8192 count=1 skip=<block_offset>
```

Restore old version of the block

```
dd if=/tmp/783-old.dmp of=/dev/sdc bs=8192 count=1 seek=<block_offset> conv=notrunc
```

Can ASM scrubbing detect and repair lost writes?

Test Scenario

1. Calculate the ASM disk offset of the primary copy of the data block
2. Extract the current version of the data block with dd
3. Update the block in the database via SQL and checkpoint it
4. Restore the old version of the data block to the primary extent
5. Scrub the affected datafile

DEMO
tc07

ASM Triple Mirroring – Single Mirror Lost Write Scenarios

Mirror 1 Write	Mirror 2 Write	Mirror 3 Write	Detected by Default?	Repaired by Default?
LOST	OK	OK	YES	NO
OK	LOST	OK	YES	NO
OK	OK	LOST	YES	NO

ASM Triple Mirroring – Multiple Mirrors Lost Write Scenarios

Mirror 1 Write	Mirror 2 Wrote	Mirror 3 Write	Detected by Default?	Repaired by Default?
OK	LOST	LOST	YES	NO
LOST	OK	LOST	YES	NO
LOST	LOST	OK	YES	NO

The scrub repair logic can detect the most recent version of a data block!

Lost Write Detection & Repair – Summary

1. **ASM scrubbing does not repair lost writes by default!**

   Enable ASM Scrub Lost Write Repair (defaults to false):

   ```
   alter system set "_asm_enable_repair_lostwrite_scrub"=true;
   ```

2. **Exadata Celldisk Scrubbing does NOT check for lost writes (it only checks disk sectors)!**

3. **Database Lost Write Protection (db_lost_write_protect) won' t help either!**

Thank you for attending!

SYMPOSIUM 42

Thank you for attending!

Questions, feedback, comments?
I look forward to hearing from you!

christoph.lutz@swisscom.com

@chris_skyflier

@christophlutz.bsky.social

GitHub Christoph-Lutz

References (1/3)

Alex Fatkulin, Consistent gets from cache (fastpath), 2009-01-14

Alex Fatkulin, Consistent gets from cache (fastpath) 2, 2009-02-07

Franck Pachot, Dirty Reads in Oracle Database (is Oracle ACID across failure?), 2022-10-21

Franck Pachot, Oracle Rolling Invalidate Window Exceeded, 2016-08-30

Franck Pachot, Oracle Rolling Invalidate Window Exceeded(3), 2021-02-24

Franck Pachot, Oracle write consistency bug and multi-thread de-queuing, 2020-12-14

Franck Pachot, READ COMMITTED anomalies in PostgreSQL , 2021-07-13

Jonathan Lewis, Correlation Oddity, 2011-12-19

Jonathan Lewis, Consistent?, 2024-12-10

Jonathan Lewis, Oracle Core: Essential Internals for DBAs and Developers, Apress, 2011

Jonathan Lewis, Redo, 2011-08-19

Hatem Mahmoud, DML restart and tracked columns, 2018-11-15

Hatem Mahmoud, Write consistency and DML restart, 2018-10-05

Hatem Mahmoud, Write consistency : Not all restarts are equals, 2018-11-08

Kun Sun, Oracle Write Consistency and "ORA-00600: [13030], [20]", 2022-04-11

References (2/3)

Kun Sun, Oracle Write Consistency and ORA-30926: "unable to get a stable set of rows in the source tables", 2022-04-11

Laurenz Albe, Comparison of the transaction systems of Oracle and PostgreSQL, 2025-02

Laurenz Albe, Transaction anomalies with SELECT FOR UPDATE, 2022-06

Luca Canali, ASM Metadata and Internals, May 2014

My Oracle Support, Bug 33470254 - UPDATE FAILS WITH ORA-00600 [13030], [20] (Doc ID 33470254.8), 2025-01-26

My Oracle Support, Complex Insert Intermittently Slower on Partitioned Table as a Result of Cursor Invalidation (Doc ID 2057107.1), 2022-08-29

My Oracle Support, Insert Statement On Partitioned Tables Is Restarted After Invalidation (Doc ID 1462003.1), 2022-03-07

My Oracle Support, Large Wait Event "Library Cache Pin" (Doc ID 2298011.1), 2019-12-11

My Oracle Support, ORA-600 [13030] (Doc ID 351760.1), 2024-11-07

My Oracle Support, Ora-00600 [13030], [20] During Update Statement Using V$ tables (Doc ID 1400439.1), 2019-03-30

My Oracle Support, Rolling Cursor Invalidations with DBMS_STATS.AUTO_INVALIDATE (Doc ID 557661.1), 2024-12-30

My Oracle Support, Troubleshooting Library Cache: Lock, Pin and Load Lock (Doc ID 444560.1), 2024-09-11

My Oracle Support, The "Execs" of SQL Monitor Report Is Larger Than The Execution Count of DML Statement Sometimes (Doc ID 2639748.1), 2024-07-20

Nigel Bayliss, Fine-Grained Cursor Invalidation, 2021-06-20

References (3/3)

Norbert Debes, Secrets of the Oracle Database, Apress, 2009

Oracle 19c, Backup and Recovery Reference, Section 3.25: VALIDATE

Oracle 19c, SQL Tuning Guide, Section 20.1.4.2: Cursors Marked Rolling Invalid

Oracle Corp., Oracle 23ai, Database Reference, Section A.2: Physical Database Limits

Randolf Geist, DML Operations On Partitioned Tables Can Restart On Invalidation , 2016-01-17

Steve Adams, Oracle8i Internal Services for Waits, Latches, Locks, and Memory, O'Reilly, 1999

Tanel Poder, TPT Github Repository, Script ashtop.sql

Tanel Poder, TPT Github Repository, Script ash_wait_chains.sql

Tanel Poder, TPT Github Repository, Script sqlmon_restarts.sql

Tanel Poder, Oracle SQL Monitoring and Write Consistency Demo, Youtube Video, 2018-09-06

Tom Kyte, Something Different Part I of III, 2005-08-30

Tom Kyte, Part II, Seeing a Restart, 2005-08-31

Tom Kyte, Part III Why Is a Restart Important to Us, 2005-09-01

Appendix A:
Write Consistency & DML Retries

Three Pass Algorithm – Internals

```
-> updexe
   -> updThreePhaseExe
```

```
   <- updThreePhaseExe
<- updexe
```

Three Pass Algorithm – Internals

```
-> updexe
   -> updThreePhaseExe
```

UNLOCKED	-> updaul updrow kddlrc kdudcp updrow kddlrc kducp <- updaul	First and optimistic attempt to update columns. If this fails due to write consistency conflicts, enter the LOCK phase.
LOCK	-> updaul updrow kddlkr updrow kddlkr <- updaul	Attempt to lock all the rows that are to be updated. If this succeeds, enter the ALL LOCKED phase. If this fails, start additional LOCK passes until one succeeds; fail with ORA-600 [13013] after 5,000 attempts.
ALL LOCKED	-> updaul updrow kddlrc kdudcp updrow kddlrc kdudcp <- updaul	Final attempt to update all rows This will succeed if all the needed rows in the previous LOCK phase can get locked. Note that this may not be possible with non-deterministic rowsets (e.g. when using rownum, non-deterministic functions or querying v$ views). In that case, this phase fails with ORA-600 [13030] ("no stable result set").

updaul:
Attempt to update or lock all rows

updrow:
Update a row

kddlrc:
Kernel Data Lock Row (with Cleanout?)

kducp:
Kernel Data Update Compare?

kddlkr:
Kernel Data Lock Row

```
   <- updThreePhaseExe
<- updexe
```

Threat – ORA-1551 Statement Restarts and Retries

ORA-1551
Undo Segment Extension

Conventional Path
updrow

No statement level restart

Row-level retry (ROW_RETRY)

Array Operations
updrowFastPath

Phase NOT LOCKED: Rollback + statement restart!

Phase LOCKED: Row-level retry (ROW_RETRY)

Fast Path disabled by:

- Triggers
- Constraints
- Partitioning
- Supplemental logging
- Possibly other factors

Write Consistency Restarts vs DML Retries – Summary

	Write Consistency Restart	DML Retry
Restart Handler Function	updThreePhaseExe	opiexe
Restarted Function	updaul (update: attempt update or lock of all rows)	updexe (update: execute)
Restart Triggers	Write conflicts	Internal errors (e.g. ORA-14403 and possibly others)
Restart Retry Limit	5,000 (lock-retry limit; hard-coded into updThreePhaseExe)	None (unlimited)
Observability	SQL Trace (str=n), SQL Monitor (starts=n) bpf: update-restarts.bt, ksesecl0-sid-err.bt	v$sysstat and v$sesstat: "DML statements retried"
SQL Exec ID	Doesn't change on restart	Incremented on every retry (changes every time opiexe calls updexe again)

Appendix B: Buffer Cache Mechanics

Oracle (R)DBA – (Relative) Data Block Address: Smallfile Tablespace

DBA Display Format

15/139

Encoding (32 bit)

00000011 11000000 00000000 10001011

File Number: 10 bit

Block Number: 22 bit

Datafile #15

Limits

Datafiles per TS: (2^10)-1 = 1023

Blocks: (2^22)-1 = 4,194,303

File Size: ~32 GB (8K Block Size)

cf. Oracle 23ai, Database Reference, Section A.2 Physical Database Limits

Oracle (R)DBA – (Relative) Data Block Address: Bigfile Tablespace

DBA Display Format

1024/62914699

Encoding (32 bit)

00000011 11000000 00000000 10001011

Block Number: 32 bit
(File number: 1024 - always fixed; not stored)

Datafile #15

Limits

Datafiles per TS: 1

Blocks: 2^32 = 4,294,967,296

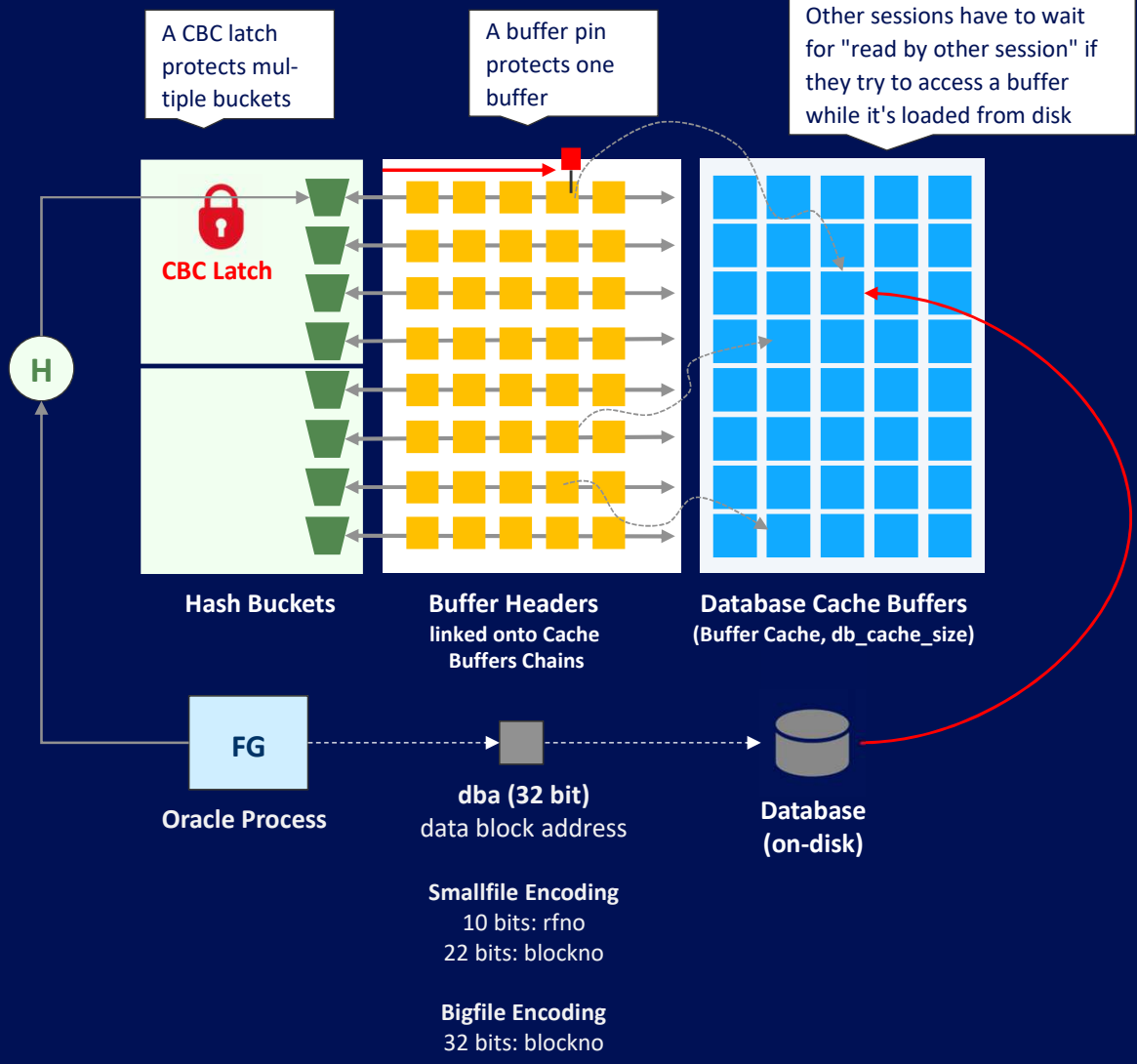
File Size: 32 TB (8K Block Size)

cf. Oracle 23ai, Database Reference, Section A.2 Physical Database Limits

Buffer Cache Mechanics

1) Lookup/identify data block address
2) Hash and map dba to bucket
3) Acquire Cache Buffers Chain (CBC) Latch
4) Walk the Cache Buffers Chain
5) Pin the buffer (buffer header)
6) Release CBC Latch
7) Read block from disk (if not in cache)
8) Access block (read, modify)
9) Acquire CBC Latch
10) Unpin the buffer (buffer header)
11) Release CBC Latch

This is an awful lot of work!



Buffer Cache Mechanics – Optimizations

consistent gets - examination

- requires only a single latch get
- generally used to read undo blocks and to access unique index blocks

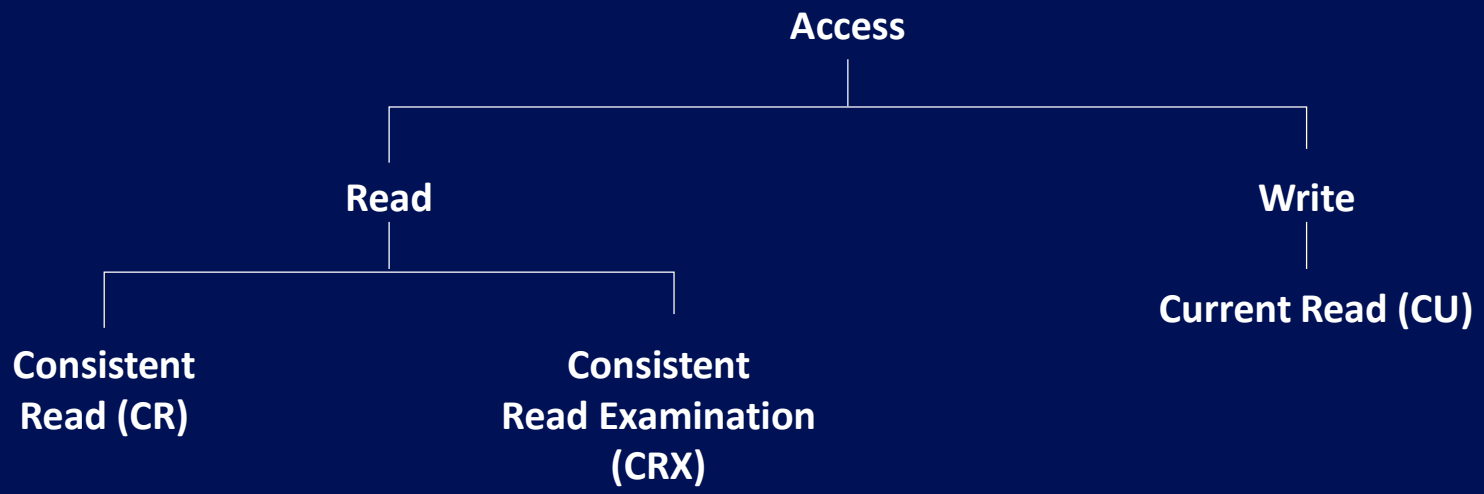
buffer is pinned count

- Oracle can hold a pin for the duration of a database call
- If a pinned buffer is revisited, Oracle doesn't have to acquire the CBC latch and this will save logical I/Os

db block gets from cache (fastpath)

- fastpath = massive LIO reduction
- Exact mechanics unknown, needs more research :-)

Buffer Cache Access – Reads and Writes



Buffer Cache Access – Consistent Read (CR) Scenarios

Consistent Read Scenarios

Block SCN <= Snapshot SCN

- Block unchanged since query started
- Read it as-is

Block SCN > Snapshot SCN

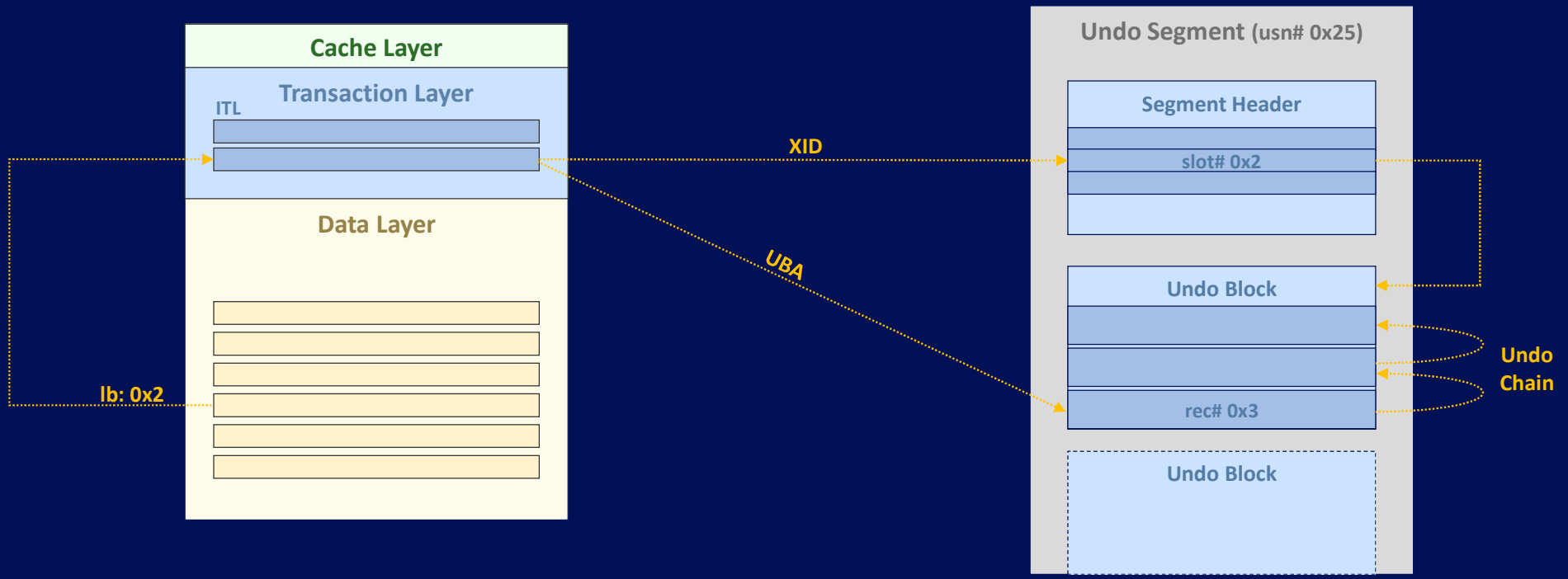
- Block changed since query started
- Create a read consistent version of the block using undo data
- apply undo records until:
  block version <= snapshot SCN

Snapshot SCN (query SCN) = SCN at the start of a query, dml statement, or transaction

Appendix C: Transactions & MVCC

Transactions & MVCC – Modification

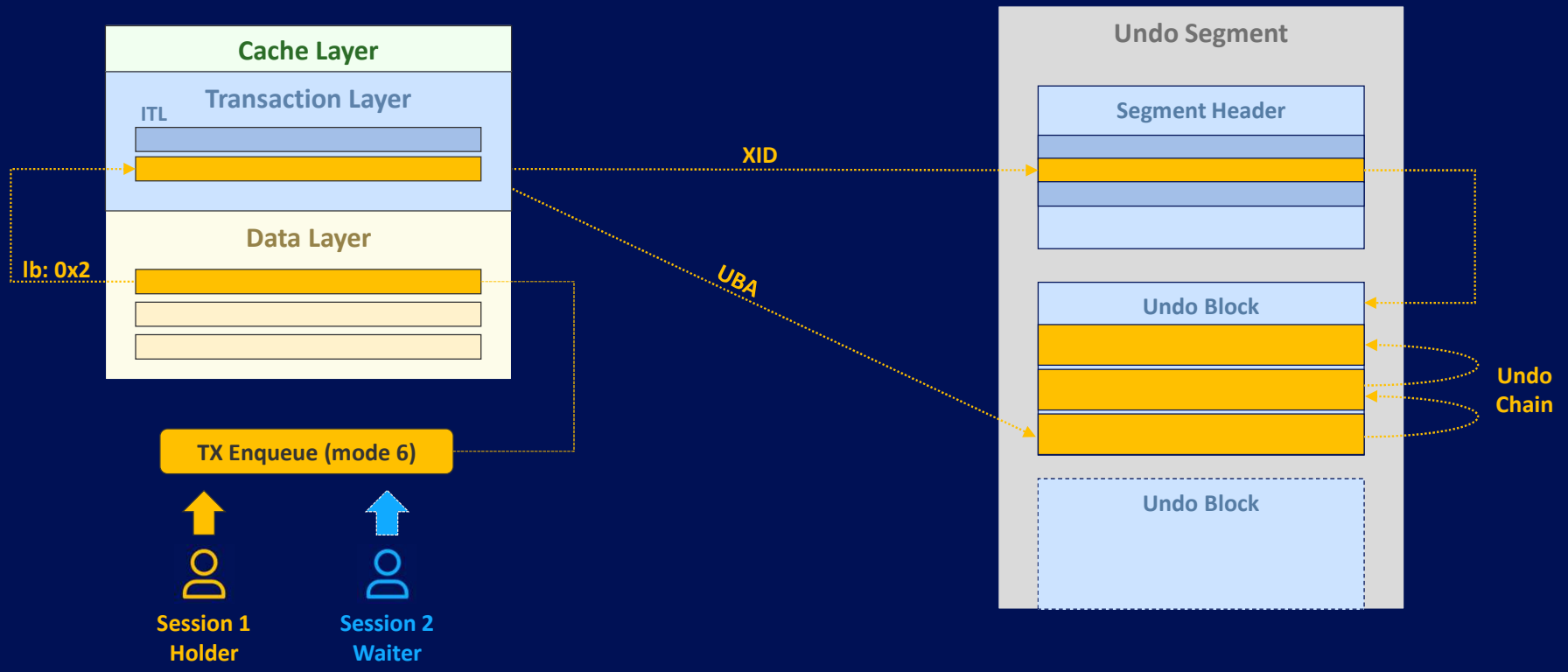
Itl	Xid	Uba	Flag	Lck
0x01	0x0020.010.00005d8a	0x00002726.1bfd.09	C---	0
0x02	0x0025.002.000091aa	0x00000582.2920.03	----	1



- Row locks are part of a database block
- Undo used for tx rollback and read consistency (CR)
- Lazy ITL slot cleanup (fast commit and delayed block cleanout)

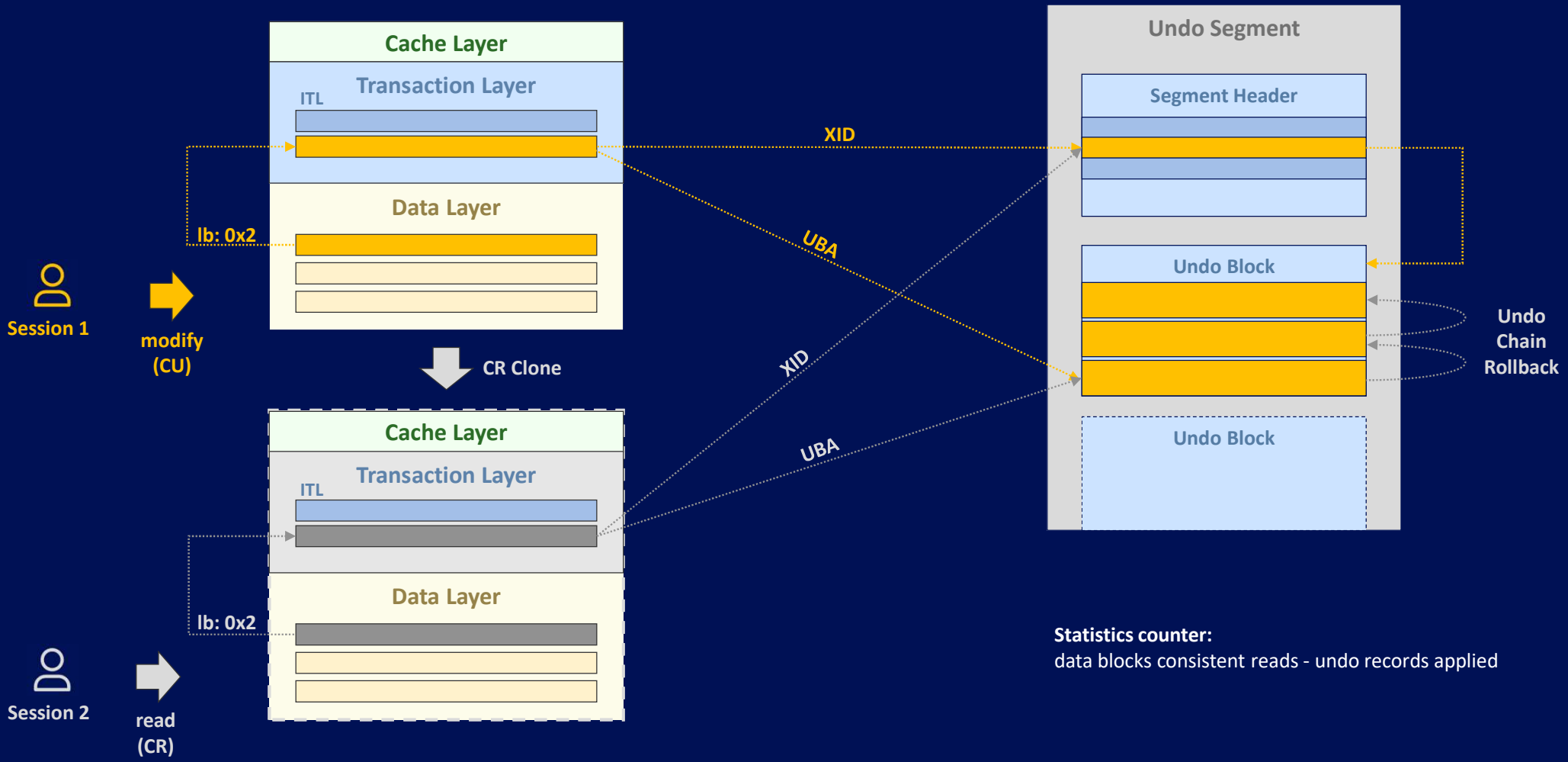
xid: usn#.slot#.wrap#
uba: dba.seq#.rec#

Transactions & MVCC – Modification: Locking Conflict



- Different sessions cannot update the same row at the same time!
- However, different sessions can update different rows at the same time (write consistency)

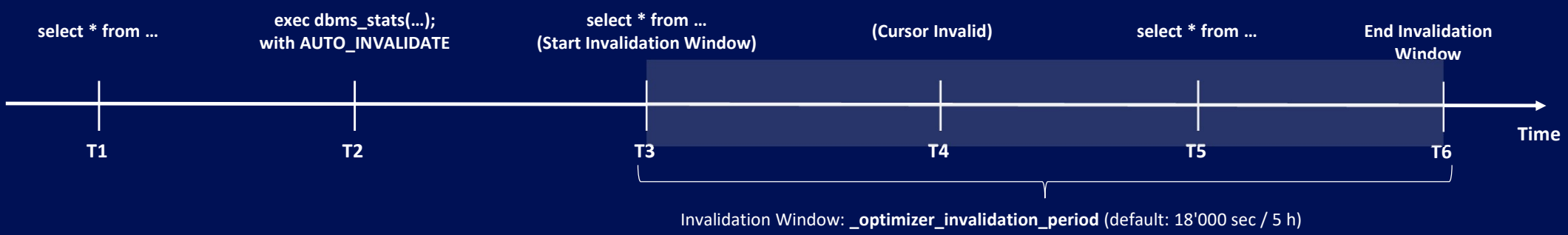
Transactions & MVCC – Read Consistency



Statistics counter:
data blocks consistent reads - undo records applied

Appendix D: Rolling Cursor Invalidation

Rolling Cursor Invalidation – Example: DBMS_STATS



Cursor parsed and loaded for the first time.

Cursor is marked "ROLLING_INVALID=Y"

Cursor is executed again and marked "ROLLING_INVALID=X'. Invalidation Window starts.

Pseudo-randomly defined point in time at which cursor becomes invalid

Cursor is hard-parsed and a new child cursor is created.

Invalidation Window ends.

Appendix E: Paranoid Concurrency Mode

Paranoid Concurrency Mode – Wait Event "log file sync - queried data"

```
SQL> select name, parameter1, parameter2, parameter3
     from v$event_name w
     here name = 'log file sync - queried data';

NAME                          PARAMETER1    PARAMETER2    PARAMETER3
----------------------------- ------------ ------------ -----------
log file sync - queried data  block#       thread scn    sync scn
```

```
SQL> select sid, serial#, event, p1, p2, p3, seq# from v$session where sid = &&sid;

      SID    SERIAL# EVENT                              P1         P2          P3
 -------- ---------- ---------------------------- ---------- -------- ----------
      267      43818 log file sync - queried data 4294967295 17793531          0
```

Always set to 0.

Always set to UB4MAXVAL, that is: power(2,32)-1

Session proceeds as soon as On Disk SCN > Thread SCN

Paranoid Concurrency Mode – struct kcrfs kcrfsg_

```
SQL> oradebug dump global_area 2

struct kcrfs kcrfsg_ [0600C8FA8, 0600C96E8) = 7AEB7430 00000000 000030FD ...
Dump of memory from 0x0600C8FB4 to 0x0600C96E8
0600C8FB0          000030FC 00064000 45717A3D      [.0...@..=zqE]
0600C8FC0 00000000 00000000 00000000 00000000  [................]
0600C8FD0 00000000 00000000 00064000 00000000  [.........@......]
0600C8FE0 00ECE423 00000000 00000000 00000000  [#...............]
0600C8FF0 000007BA 00000000 00000000 00000000  [................]
0600C9000 00000000 00000000 00127400 00000000  [.........t.`....]
0600C9010 00ECE423 00000000 00ECE423 00000000  [#.......#.......]
0600C9020 00ECE423 00000000 00000013 000030FD  [#............0..]
0600C9030 AF530010 0000078A 00000000 00000000  [..S.............]
0600C9040 60127580 00000000 00000000 00000000  [.u.`............]
0600C9050 00000000 00000000 00000000 00000000  [................]
0600C9060 00ECE422 00000000 00000000 00000000  ["...............]
0600C9070 00000000 00000000 00000000 00000000  [................]
0600C9080 00000000 00000000 60129B00 00000000  [............`....]
0600C9090 45717A3D 00000000 C0438C00 00000000  [=zqE......C.....]
0600C90A0 000002BD 00000000 00000789 00002155  [............U!..]
0600C90B0 000000D3 00000000 000CB320 00000000  [........ .......]
0600C90C0 000001F0 00000000 00000200 00000000  [................]
0600C90D0 00000800 00000013 00000001 00000000  [................]
 ...
```

On Disk SCN value

On Disk SCN SGA address

Paranoid Concurrency Mode – Foreground Process Code Paths

Session 1 – COMMIT

```
semtimedop ()
sskgpwwait ()
skgpwwait ()
ksliwat ()
kslwaitctx ()
kcrf_commit_force_int ()
ksupopg ()
opiodr ()
ttcpip ()
opitsk ()
opiino ()
opiodr ()
opidrv ()
sou2o ()
opimai_real ()
ssthrdmain ()
main ()
```

Session 2 - SELECT

```
semtimedop () from /lib64/libc.so.6
sskgpwwait ()
skgpwwait ()
ksliwat ()
kslwaitctx ()
kcrf_commit_force_int ()
ktr_resolve_session_commit_dependencies ()
opiodr ()
ttcpip ()
opitsk ()
opiino ()
opiodr ()
opidrv ()
sou2o ()
opimai_real ()
ssthrdmain ()
main ()
```

New in 23c

With 23c Paranoid Concurrency Mode, COMMIT and SELECT both use the same code internally!

Appendix F: ASM

ASM Metadata – X$KFFXP (Kernel File eXtent Pointers) Example

Disk Group# | Disk# | File# | Progressive Extent# | Virtual Extent# | Logical Extent# | Allocation Unit#

GROUP_KFFXP	DISK_KFFXP	NUMBER_KFFXP	PXN_KFFXP	XNUM_KFFXP	LXN_KFFXP	AU_KFFXP	AU_SIZE	AU_SIZE_MB
1	25	296	0	0	0	6703		
1	11	296	1	0	1	6704		
1	4	296	2	0	2	6686		
1	8	296	3	1	0	6693	4194304	4
1	4	296	4	1	1	6687	4194304	4
1	16	296	5	1	2	6678	4194304	4
1	27	296	6	2	0	6706	4194304	4
1	2	296	7	2	1	6687	4194304	4
1	18	296	8	2	2	6692	4194304	4
1	37	296	9	3	0	6621	4194304	4
1	30	296	10	3	1	6687	4194304	4
1	15	296	11	3	2	6687	4194304	4

...

Let's walk through an example based on db file# 34, block 783

0 = Primary Extent
1 = Mirror Extent
3 = Second Mirror Extent

Map DB Block to ASM Virtual Extent (Block 37/783, Primary Extent):

floor(block# * db_block_size / AU_SIZE) = floor(783 * 8192 / 4194304) = **1**

Map DB Block to ASM Disk Offset:

((AU_KFFXP * AU_SIZE) + (block# * db_block_size) - (XNUM _KFFXP * AU_SIZE)) / db_block_size =
((6693 * 4194304) + (783 * 8192) - (1 * 4194304)) / 8192 = **3427087**

Formula source: Norbert Debes, Secrets of the Oracle Database, p. 124, Apress, 2009

Exadata Complications – cellutil Examples (1/2)

```
$ cellutil -d /dev/sdl -c primary -s header

Metadata Copy: primary
Section: CellDiskHeader
cdiskHdrSign:          o r a c l e   s a g e d i s k
asmlibMask:
checksum: 1483677998
portInfo: 0
mdVersion: 0
ossCompatibility: 0
mdSubVersion: 12
sectorSize: 512
cdiskInfoSize: 4
cdiskInfoOffset: 1
```

```
$ cellutil -d /dev/sdl -c primary -s info

Metadata Copy: primary
Section: CellDiskInfo
cdInfoSign_: * c d I n f o *
cdiskGuid_: 4eb13664-e292-407d-a1fc-f3e4f14bb937
cdInfoCheckSum_: 662617632
cdiskFlags_:
updateSeqNumber_: 336
numberGridDisks_: 6          # grid disks are segmented
gdiskExtentSize_: 32768      # in <sectorSize>
metadataSize_: 32768
cdiskSize_: 26787086336      # in <sectorSize>
gdiskSegmentMapStart_: 5
gdiskSegmentMapSize_: 1
gdiskTableRecordSize_: 5
gdiskTableStart_: 32738
cdiskState_: CDSTATE_NORMAL
cdVirtualSlot_: 11
cdModificationTimeStamp_: Sat Apr 13 00:00:14 2024
cdModificationOperation_: altGD_RECO_CD_11_stdm01
cdPmemDSC_: 0
cdiskName: C D _ 1 1 _ s t d m 0 1 c e l 0 4 _ m
cdiskDescription:
spares:
```

Exadata Complications – cellutil Examples (2/2)

```
$ cellutil -d /dev/sdl -c primary -s gdtable

Metadata Copy: primary
Section: GDTableRecord
signature: G D t a b l e !
gdiskGuid: 896387f6-62a6-4eaa-b725-cca39436a9e0
gdiskSize: 572224       # in <gdiskExtentSize>
virtualSize: 572224     # in <gdiskExtentSize>
gdiskSegments: 2
gdiskId: 4
gdiskIncarnationNumber: 4
creationTimeStamp: Wed Mar 31 06:42:03 52094
gdiskGroupName: r o o t
checkSum: 2491673749
modificationTimestamp:
gdiskState: GDISK_ACTIVE
…
RTVersionNumber: 0
RTStatus: 0
List of requiredFlash4UWIDs:
lastScrubTS: Sat May  4 06:40:04 56250
gdiskName: D A T A _ C D _ 1 1 _ …
gdiskDesc: C l u s t e r   s t d m 0 1 …
gdiskSPName:
gdiskSecurityInfo:
spares:
```

```
$ cellutil -d /dev/sdl -c primary -s segmap

SegmentMapRecord 3:
gdiskId: 4
startExtent: 2
segmentSize: 1278
gdiskOffset: 0

SegmentMapRecord 4:
gdiskId: 4
startExtent: 1281
segmentSize: 570946
gdiskOffset: 1278
```

First gdisk segment

Second gdisk segment

Refer to Appendix G for a step-by-step offset calculation example!

Appendix G: Trace Events

DML Trace

DML Tracing

```
-- Enable tracing
alter session set events 'trace[dml] disk=highest';

-- Disable tracing
alter session set events 'trace[dml] off';
```

DML and SQL Tracing Combined

```
-- Enable tracing
alter session set events 'trace[dml] disk=highest: sql_trace wait=true';

-- Disable tracing
alter session set events 'trace[dml] off : sql_trace off';
```

Trace Event sql_monitor – Force SQL Monitoring

Force SQL Monitoring

```
-- Force SQL Monitoring system-wide for sql 768j5fa2vwvu7
alter system set events 'sql_monitor [sql: 768j5fa2vwvu7] force=true';

-- Disable SQL Monitoring for sql 768j5fa2vwvu7
alter system set events 'sql_monitor [sql: 768j5fa2vwvu7] off';

-- Force SQL Monitoring 5 times only (end after 5 occurences)
alter system set events 'sql_monitor [sql: 768j5fa2vwvu7] {occurence:end_after 5} force=true';

-- Force SQL Monitoring for multiple sql statements
alter system set events 'sql_monitor [sql: 5hc07qvt8v737|sql: 9ht3ba3arrzt3] force=true';

-- Disable SQL Monitoring multiple sql statements
alter system set events 'sql_monitor [sql: 5hc07qvt8v737|sql: 9ht3ba3arrzt3] off;
```