Can we survive Oracle data block corruption

NoCoug

Aug 21, 2014

Lorrie Yang

About the Presenter

- In eCommerce DBA team at Bank of America
- Supporting Oracle databases since oracle 7
- Currently supporting
  - OLTP databases with high load
  - RAC
  - Active DataGuard
  - Multi-way GoldenGate replication

Scope of this presentation

- Database or ASM corruptions are not covered
  - Lost data files
  - Lost redo logs
  - Lost control files
  - ASM issues
- Data Block Corruption – focus of this presentation
  - Data block is a generic term, it refers to all oracle blocks including table, index, undo, data file header, segment header, control file header, etc.
  - Block Corruption=Failure of block(s) consistency check.

Audience Survey

Content

- Block Corruption Causes
- Block Structure
- Block Corruption Types
- Block Corruption Detection
  - In time
  - On going
- Block Corruption Verification
- Block Corruption Repair
  - Strategies
  - Techniques
- Examples

Block Corruption Causes

- Hardware - faulty disk and disk controllers, firmware.
- Hardware - faulty memory, caching problem.
- Software - oracle database software defects

Oracle Block Structure

- Cache Layer
  - In block header common area
- Transaction layer
  - In block header
  - Fixed
  - variable
- Data layer
  - Table directory
  - Row directory
  - Free space
  - Row data
- Footer/tail
  - Last 4 bytes of a block

Block Structure – header Cache Layer

- Total 20 bytes
- Block Type (table, index, undo, temp) (typ). 1 byte.
- Block format (frmt). 1 byte.
- Filler/unused. 2 bytes.
- Relative Data Block Address (rdba). 4 bytes
- SCN (SCNbase, SCNwrap). 4+2 bytes
- Number of changes made by same SCN (Seq). 1 byte
- Flg. 1 byte
- CheckSum (Chkval). 2 bytes
- Filler/reserved. 2 bytes.

Block Structure – header Cache Layer-CheckSum

- Is a number calculated from all the bytes in the block.
- Normally added into a block right before the block is written to disk
- Data block checksum is done by DBWR
- Verified at read time (DB_BLOCK_CHECKSUM in TYPICAL or FULL mode)
- Log block checksum is done by foreground process and/or LGWR.

Block Structure - footer

- Total 4 bytes
- Contains lower order two bytes of SCNbase, block type, SCN seq number.
- Block version (SCNbase, seq number, block type) must be consistent with same data in cache layer.

Block Dump

- OS/Physical dump
  - Directly from disk. DB Online is ok.
  - dd bs=db_block_size if=dbfile.dbf skip=#blocks_to_skip count=#blocks_to_dump | od –xv > dumpfile.txt
  - Must understand data block layout.
  - Offset in octal format. Data in hex. Hard to decode.
- Oracle/Logical dump
  - SQL> alter system dump datafile # block #; -- from memory
  - SQL> alter system dump datafile 'filename' block #; -- from disk

OS block dump output example

```
0000000 0000 0000 0000 0000 0000 0000 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
0000040 0000 0000 0000 0000 0000 0000 0000 0000
0000060 0000 0000 0000 0000 0000 0000 0000 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
0000120 0000 0000 0000 0000 0000 0000 0000 0000
0000140 0000 0000 0000 0000 0000 0000 0000 0000
0000160 0000 0000 0000 0000 0000 0000 0000 0000
0000200 0000 0000 0b01 0000 a21d 0000 0002 0040
0000220 bd4c 0022 0000 0402 65a2 0000 0001 0000
0000240 0008 0000 f400 0001 0009 0000 3200 0000
0000260 0000 000a 007e 0000 f3ff 0001 1260 0000
0000300 2bdf 0000 4160 000b 0000 0000 0000 0000
0000320 0000 0000 0000 0000 9300 0000 0080 0000
0000340 0000 0000 0000 0000 0000 0000 0000 0000
0000360 0000 0000 0000 0000 0000 0000 0000 0000
0000400 0000 0000 0000 0000 0000 0000 0000 0000
…
0017620 0000 0000 0000 0000 0000 0000 0000 0000
0017640 0000 0000 0000 0000 0000 0000 0000 0000
0017660 0000 0000 0000 0000 0000 0000
0017674
```

Oracle Formatted Block Memory Dump - example of a good block

```
Start dump data blocks tsn: 5 file#:5 minblk 12 maxblk 12
Block dump from cache:
Dump of buffer cache at level 4 for tsn=5 rdba=20971532
Block dump from disk:
buffer tsn: 5 rdba: 0x0140000c (5/12)
scn: 0x0000.00032e89 seq: 0x01 flg: 0x04 tail: 0x2e891e01
frmt: 0x02 chkval: 0x8191 type: 0x1e=KTFB Bitmapped File Space Bitmap
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00002B5513EEAA00 to 0x00002B5513EECA00
2B5513EEAA00 0000A21E 0140000C 00032E89 04010000  [......@.........]
2B5513EEAA10 00008191 00000005 0045C080 00000000  [..........E.....]
2B5513EEAA20 00000000 0000F800 00000000 00000000  [................]
2B5513EEAA30 00000000 00000000 00000000 00000000  [................]
      Repeat 507 times
2B5513EEC9F0 00000000 00000000 00000000 2E891E01  [................]
File Space Bitmap Block:
BitMap Control:
RelFno: 5, BeginBlock: 4571264, Flag: 0, First: 0, Free: 63488
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
...
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
2014-08-18 17:00:12.908070 : kjbmbassert [oxc.5]
2014-08-18 17:00:12.908170 : kjbmsassert(oxc.5)(to 2)(lvl 4)
End dump data blocks tsn: 5 file#: 5 minblk 12 maxblk 12
```

Block Consistency Check within a block

- Performed when a block is read from disk
  - Block version between block header and block footer
  - Rdba value
  - Block checksum calculated vs. stored in block.
  - Detect disk IO corruption.
- Performed after data change in memory when DB_BLOCK_CHECKING is TRUE.
  - Detect memory corruption

Data Block Corruption Types

- Physical corruption vs. logical corruption
- Intrablock vs. interblock corruption

Data Block Corruption Type
Physical vs. Logical

- Physical/media corruption
  - block checksum mismatch at read time vs. already written in the block.
  - All zeros in a block
  - SCN=0 (zero)
  - Header and footer mismatch/Block version mismatch in a block.
- Logical/Software corruption
  - Block does not appear physical corrupted but content doesn't make logical sense.
  - Blocks contents mismatch such as between a table block and its index block.

Data Block Corruption Type intrablock vs interblock

- Interblock corruption
  - Can be physical or logical
  - Can be detected by all utilities
  - Is reported in alert log and v$ views, and trace files
  - Can be fixed in multiple ways
- Intrablock corruption
  - Can only be logical
  - Can only be detected by dbverify and analyze command
  - Tracked in ADR only
  - Can only be fixed by renewing (drop, rebuild)

Data Block Corruption Detection

- Prevention – is it possible ? Sadly no.
- Detect as early as possible – Why is it very important?
- In-time detection
- Regular detection

Data Block Corruption Detection in time detection

- Detect as soon as corruption happens
  - WARNING !! Overhead of 1-10% of ongoing impact.
  - 11g DB parameter
    - DB_ULTRA_SAFE
  - Before 11g DB parameter. More flexible.
    - DB_BLOCK_CHECKING
    - DB_BLOCK_CHECKSUM
    - DB_LOST_WRITE_PROTECT
- Alert as soon as your alert log monitoring

Data Block Corruption Detection in time detection continued

parameter	value	overhead	Recommended/Default	comment
DB_BLOCK_CHECKING	FULL/TRUE	1-10%	Recommended on standby	Block integrity check on all
	MEDIUM			Block integrity check on all except indexes
	LOW			Basic block header check only after data change in memory
	OFF/FALSE		default	Block integrity check for system tablepsace only before written to disk
DB_BLOCK_CHECKSUM (10gR2 and above)	FULL/TRUE	4-5%	Recommended on standby	Detect both memory and disk corruption
	TYPICAL	1-2%	default	Detect disk corruption
	OFF/FALSE			Checksum checked at read time only for system tablepsace

Data Block Corruption Detection in time detection continued

parameter	value	overhead	Recommended/Default	comment
DB_LOST_WRITE_PROTECT	FULL	low	Recommended on both primary and standby	Detect write loss for all tablespaces
	TYPICAL			Detect write loss for all tablespaces except read only tablespaces
	NONE		Default	No lost write detection

Data Block Corruption Detection in time detection continued

parameter	value	comment
_DB_LOCK_CHECK_FOR_DEBUG	TRUE	Check more and dump block before image. Usually set for root case analysis in a test system.
DB_ULTRA_SA FE (11g only)	DATA_AND_INDEX	DB_BLOCK_CHECKING = FULL. DB_BLOCK_CHECKSUM = FULL. DB_LOST_WRITE_PROTECT = TYPICAL.
	DATA_ONLY	DB_BLOCK_CHECKING = MEDIUM. DB_BLOCK_CHECKSUM = FULL. DB_LOST_WRITE_PROTECT = TYPICAL.
	OFF	This is default

Block Corruption Detection
Regular detection - utilities

- Dbverify utility
- Analyze command
- Full export
- Rman

Block Corruption Detection Regular detection- deverify

- Doc ID 35512.1
- dbv file=/ora01/oradata/system01.dbf blocksize=8192
- Only on data files, not archive logs
- Checks block consistency
- No data-index cross checks
- Can run on primary and standby

Block Corruption Detection Regular detection- Analyze

- Analyze commands
  - Analyze table xxx validate structure cascade online;
  - Analyze index yyy validate structure online;
  - For partitioned tables, use a different command to validate a row belongs to a partition.

    Analyze table xxx partition ppp validate structure cascade into invalid_rows;
- Runs at object level
- Validate table-index entries/cross object/block integrity

Block Corruption Detection
Regular detection- expdp

- expdp system/ppppp full=y log=db_check.log file=/dev/null
- No need for an actual dump file
- Performs full table scan to high water mark on all tables
- Read only partial dictionary info needed for export but not all
- Does not detect all corruptions, such as
  - disk corruption above table high water mark
  - Index corruption
  - Free or temporary extent corruption
  - Corruption in system tablespace not read by the expdp process.

Block Corruption Detection Regular detection- Rman

- Detect physical corruption by default. NOCHECKSUM is off.
- Check corruption in data files, archive logs and control files.
- "Backup check logical full database/archivelogs".
  - Check both logical and physical corruption. Highly recommend.
  - Alternatively, "validate check logical database" in 11g or "Backup validate check logical database" before 11g

Block Corruption Monitoring

- DB alert logs
- v$backup_corrupton and v$database_block_corruption (prefered)

Block Corruption Monitoring – DB alert log

- ORA-01578: ORACLE data block corrupted (file # 7, block # 3)
- ORA-600 [kddummy_blkchk]
- ORA-1498 "block check failure - see trace file“
- ORA-600: internal error code, arguments: [6033], [], [], [], [], [], [], []
- ORA-8103 “object no longer exits”
- ORA-1499 "table/index cross reference failure”
- Etc.

Block Corruption Monitoring - v$backup_corruption

SQL> desc v$backup_corruption

```
Name                                      Null?    Type
----------------------------------------- -------- -------------
RECID                                              NUMBER
STAMP                                              NUMBER
SET_STAMP                                          NUMBER
SET_COUNT                                          NUMBER
PIECE#                                             NUMBER
FILE#                                              NUMBER
BLOCK#                                             NUMBER
BLOCKS                                             NUMBER
CORRUPTION_CHANGE#                                 NUMBER
MARKED_CORRUPT                                     VARCHAR2(3)
CORRUPTION_TYPE                                    VARCHAR2(9)
```

- Only report corrupt blocks related to backup.

Block Corruption Monitoring - v$database_block_corruption

SQL> desc v$database_block_corruption

```
Name                                      Null?    Type
----------------------------------------- -------- -------------
FILE#                                              NUMBER
BLOCK#                                             NUMBER
BLOCKS                                             NUMBER
CORRUPTION_CHANGE#                                 NUMBER
CORRUPTION_TYPE                                    VARCHAR2(9)
```

Displays blocks marked corrupt by various tools – Rman, dbverify, analyze.

Block Corruption Verification

- Identify the scope of corruption
  - Hardware issue turns to cause more than one block corruption.
  - Find out if there are more corruptions than already reported. (dbv, Rman)
- Locate the object(s)
  - Clue is in trace file with object name, or file # and block#
- Confirm the object is corrupted
  - Analyze table/index validate structure
  - Tips
    - Consider use online option
    - Consider run it on your standby
    - May need run it a few times to confirm the corruption is consistent.
    - Bounce the database then check again also for corruption consistency.

Block Corruption Repair Strategies

- Identify the object type.
  - Table? Index?
  - This leads to using different techniques to repair.
- Identify smallest unit.
  - Partition? Subpartition ?
  - This can reduce the scope of repair effort significantly.
- Identify corruption type.
  - Physical? Logical?
  - This prevents wrong techniques are used in repair thus saving valuable time.
  - If physical and cause is found, fix any hardware failures first before attempting any DB repair.
- Identify all indexes on that table
  - in case you need rebuild the indexes later.
- Consider if you can afford losing data on those data block(s).
  - This takes the repair to a completely different angle. No need to rebuild the table nor restore data.
- Save data in that corrupted table if you can/need.
  - Skip the blocks, then export or select from the table.
- Perform block dump before and after repair
  - For analysis later

Block Corruption Repair Techniques

- Active DataGuard
  - fix physical corruption automatically.
- Restore from Rman backup
  - fix only physical corruption !!!
  - Make a dump of the file/blocks before/after restore/recover.
  - “blockrecover corruption list” recover all blocks listed in v$database_backup_corruption
  - “recover datafile # block #” recover one block only
  - If recover to the time before corruption happened, partial data loss !
- Block renewing/rebuild
- Mark a block unusable/corrupted
  - partial data loss!
  - Use dbms_repair package
  - Blocks marked corrupt are automatically skipped during index and table scans.

Block Corruption Repair Summary

- Index
  - Restore from Rman (physical only)
  - Rebuild (either physical or logical)
- Table
  - Restore from Rman (physical only)
  - If good export dump file exists. Rebuild table. Import.
  - No good export dump file exists, skip corrupted blocks at new export, rebuild table then import.
  - Use SQL, PL/SQL to pull data into a new staging table by not including corrupted rows (rowid) then rebuild and restore data from staging table/renaming staging table.

Disclamation

- Always work with oracle support for data repair.
- All examples are provided for presentation purpose. Neither the presenter nor Bank of America should be held responsible for your data loss resulted from trying any techniques mentioned in this presentation.
- Schema names and table names have been edited for presentation purpose.

Active DataGuard automatic repair physical corruption example

In 11.2.0 DB alert log:

WARNING: AutoBMR fixed mismatched on-disk block cbc7fe65 with in-mem rdba 20bfa6c.

Sat Jun 28 07:42:45 2014

Automatic block media recovery successful for (file# 8, block# 785012)

Automatic block media recovery successful for (file# 8, block# 785012)

WARNING: AutoBMR fixed mismatched on-disk block cbc7fe65 with in-mem rdba 20bfa74.

Automatic block media recovery successful for (file# 8, block# 785013)

WARNING: AutoBMR fixed mismatched on-disk block cbc7fdec with in-mem rdba 20bfa75.

Corrupted table/table subpartition example

- The pain without early detection. There was no in-time nor regular detection.
- Logical corruption
- Recover with data loss

Corrupted table/table subpartition example – early symptom

- Daily dbms_stats job suddenly failed on 11/30 with limited and misleading info.
- Support case was open with Oracle support right away but nothing came out obvious.

11/28 1AM GMT – stats job ran 50 min successfully
11/29 1AM GMT – stats job ran 30 min successfully
11/30 1AM GMT – stats job failed with "insufficient privilege" after running for 2hour 24 min.
12/01 1AM GMT- stats job failed with "insufficient privilege" after running for 3hour 24 min.
12/02 1AM GMT – stats job had error "ORA-03113 end of file on communication channel " after running for 2 hour 7 min. with ora-7445 and ora-600 in alert log
12/03 1AM GMT – stats job had error "ORA-03113 end of file on communication channel" after running for 1 hour 43 min. with ora-7445 and ora-600 in alert log
12/04 1AM GMT – stats job had error "ORA-03113 end of file on communication channel" after running for 1 hour 47 min. with ora-7445 and ora-600 in alert log

- One sample of stats job output:

```
Tue Dec  4 01:00:01 GMT 2012 gather_stats.ksh with process ID 6952 starts
.
BEGIN dbms_stats.gather_database_stats(options=> 'GATHER AUTO'); END;

*
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 7275
Session ID: 1143 Serial number: 6057

 Query to database failed with error

Tue Dec  4 02:47:35 GMT 2012 sending email alert
```

Corrupted table/table subpartition example – early symptom continued

- Two days later, ORA-7445 and ORA-600 started appear in DB alert log

Exception [type: SIGSEGV, Address not mapped to object] [ADDR:0x2AF593027000] [PC:0x40B9290, __intel_new_memcpy()+5424] [flags: 0x0, count: 1]

Errors in file /oracle/diag/rdbms/inst1/INST1/trace/INST1_ora_11079.trc (incident=604777):

ORA-07445: exception encountered: core dump [__intel_new_memcpy()+5424] [SIGSEGV] [ADDR:0x2AF593027000] [PC:0x40B9290] [Address not mapped to object] []

Incident details in: /oracle/diag/rdbms/inst1/INST1/incident/incdir_604777/INST1_ora_11079_i604777.trc

.

.

ORA-00600: internal error code, arguments: [kgh_heap_sizes:ds], [0x2AF58E63E2D8], [], [], [], [], [], [], [], [], [], []

ORA-07445: exception encountered: core dump [penallnn()+168] [SIGSEGV] [ADDR:0x0] [PC:0x7FBFC30] [Address not mapped to object] []

ORA-07445: exception encountered: core dump [__intel_new_memcpy()+5424] [SIGSEGV] [ADDR:0x2AF593027000] [PC:0x40B9290] [Address not mapped to object] []

Corrupted table/table subpartition example – early symptom continued

- On same day of DB stats job failure, application batch job failed but it was not reported to DBA until one week later when application began suspecting data problem.

```
SQL> select * from oltp.event where event_id = ‘e123456'
  2  /
ERROR:
ORA-01801: date format is too long for internal buffer


no rows selected

SQL> select EFN_FLASH from oltp.event where event_id = ‘e123456';

EFN_FLASH
----------

 it should have output values.
```

Corrupted table/table subpartition example – identification

- Data issue was passed to Oracle support, the SR was then transferred to corruption team.
- Analyze command was run

  Analyze table oltp.event validate structure cascade online ;
- failed with ORA-01498: block check failure - see trace file.

Corrupted table/table subpartition example – identification continued

- *dbv was clean except for one block.*

DBVERIFY - Verification starting : FILE = +SHARED_DATA_DG01/dbname/datafile/data.363.783671577
kdrchk: Row piece pointing to itself
dba = 0x1452adca slot = 6
*Block Checking: **DBA = 340962762**, Block Type = KTB-managed data block*
data header at 0x2b1ce0756of4
kdbchk: bad row tab 0, slot 6
Page 1224138 failed with check code

Corrupted table/table subpartition example – identification continued

- Identify corrupted blocks:

SQL> select * from v$database_block_corruption ;

```
FILE# BLOCK#  BLOCKS     CORRUPTION_CHANGE# CORRUPTIO
----  ------  ---------- ----------         ------------------ ---------
81    1224138 1          2751930598         CORRUPT
```

SQL> select dbms_utility.data_block_address_file('340962762') from dual;

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE('340962762')
-------------------------------------------------
81
```

SQL> select dbms_utility.data_block_address_block('340962762') from dual;

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK('340962762')
--------------------------------------------------
1224138
```

Corrupted table/table subpartition example – identification continued

- Identify the corrupted object

```
19:23:29  SQL> select owner, segment_name, partition_name, segment_type
19:23:37    2 FROM dba_extents
19:23:41    3 WHERE file_id = 81
19:23:44    4 AND 1224138 BETWEEN block_id AND block_id + blocks - 1 ;

OWNER SEGMENT_NAME PARTITION_NAME SEGMENT_TYPE
------------------ ------------------------------- ----------------------------- -------------
-----
OLTP  EVENT EVENT_P_20121201_I3 TABLE SUBPARTITION

1 row selected.

19:30:12 SQL> SELECT PARTITION_NAME, SUBPARTITION_NAME, BLOCKS, NUM_ROWS, LAST_ANALYZED
19:35:10 2 FROM DBA_TAB_SUBPARTITIONS
19:35:13 3 WHERE TABLE_OWNER = 'OLTP' AND TABLE_NAME='EVENT'
19:35:16 4 AND SUBPARTITION_NAME = 'EVENT_P_20121201_I3';

PARTITION_NAME SUBPARTITION_NAME BLOCKS NUM_ROWS LAST_ANALYZED
------------------------------ ---------------------------------------- ---------- ---------- --
--------------------------
EVENT_P_20121201 EVENT_P_20121201_I3 0 0 02-NOV-2012 01:00:03

1 row selected.

19:42:05 SQL> select count(*) from OLTP.EVENT SUBPARTITION (EVENT_P_20121201_I3);

COUNT(*)
----------
3668309

1 row selected.

Elapsed: 00:02:59.05
```

Corrupted table/table subpartition example - repair

- Repair from Rman backup (Block Media Recovery) is suitable for physical corruption but was not suitable in this logical corruption case thus the attempt failed miserably and wasted a lot of time.

- Timing is key even when recover from Rman is suitable since a good Rman backup and subsequent archive logs must be available to fix a physical corruption by Rman.

Corrupted table/table subpartition example – repair continued

- What type of corruption was it?
- Evidence it's logical corruption
  - Daily Rman backup never failed even without check logical.
  - Active DG didn't report any automatic data fix.
  - *kdrchk: Row piece pointing to itself . Doc ID 8720802.8*

Corrupted table/table subpartition example – repair continued

- Subpartition is used in the example, but steps work also for table or partitions.
- Repair steps for logical corruption - *extract what data you can from the affected block and rebuild. Doc ID 556733.1*
  - Skip corrupted blocks
  - If the subpartition doesn't need be rebuilt, like in the case of date range partition it will be dropped after a while. The rest steps can be skipped.
  - Expdp data from that subpartition to save data from good blocks. Data from corrupted blocks will not be exported.
  - Truncate the subpartition
  - Impdp the data
  - Rebuild any indexes that may have become unusable.
  - Remove the skip flag

Corrupted table/table subpartition repair example – step 1

- Mark corrupted data blocks skipped
- This is similar to setting event 10231 (Doc ID 33405.1)
- This won't run long even for a big table if number of corrupted table blocks identified is small.

```
BEGIN
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
SCHEMA_NAME => 'OLTP',
OBJECT_NAME => 'EVENT',
OBJECT_TYPE => dbms_repair.table_object,
FLAGS => dbms_repair.skip_flag);
END;
/
```

Corrupted table/table subpartition repair example – step 2

- Must first have marked corrupted table blocks skipped.
- Save data in the corrupted table/subpartition
- Sample expdp.par file

```
dumpfile =
/ora02/DBNAME/backup/expdp_event_20121201_I3.dmp
logfile = /ora02/DBNAME/expdp_event_20121201_I3.log
content = DATA_ONLY
exclude = REF_CONSTRAINT,GRANT,STATISTICS,INDEXES
tables = OLTP.EVENT:EVENT_P_20121201_I3
status = 300
estimate = blocks
parallel = 4
```

Corrupted table/table subpartition repair example – step 3

- Truncate the table/subpartition

```
ALTER TABLE OLTP.EVENT
TRUNCATE SUBPARTITION EVENT_P_20121201_I3 update global
 indexes;
```

Corrupted table/table subpartition repair example – step 4

- Import the data back
- Sample impdp par file

```
dumpfile = /ora02/DBNAME/backup/expdp_event_20121201_I3.dmp
logfile = /ora02/DBNAME/backup/impdp_event_20121201_I3.log
content = DATA_ONLY
exclude = REF_CONSTRAINT,GRANT,STATISTICS,INDEXES
  tables = OLTP.EVENT:EVENT_P_20121201_I3
status = 300
estimate = blocks
parallel = 4
```

Corrupted table/table subpartition repair example – step 5

- Rebuild indexes that might have become unusable

```
select owner, index_name, index_type, status, table_name , table_owner
 from dba_indexes
 where
 table_owner = ‘OLTP' and
 table_name = 'EVENT' and
 status = 'UNUSABLE';
```

Corrupted table/table subpartition repair example – step 6

- Remove the skip flag

```
execute
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS(SCHEMA_NAME=>‘OLTP',
OBJECT_NAME=>'EVENT',flags=>sys.dbms_repair.NOSKIP_FLAG);
```

Q & A