

Redo internals and tuning by redo reduction

By
Riyaj Shamsudeen

Who am I?

- ❑ 15 years using Oracle
 - ❑ Over 12 years as Oracle DBA
 - ❑ Certified DBA versions 7.0,7.3,8,8i &9i
 - ❑ Specializes in performance tuning and Internals
 - ❑ Currently working for JCPenney
 - Assortment planning & Allocation systems
 - ❑ Adjunct faculty – El Centro College
 - ❑ Adjunct faculty – Northlake College
 - ❑ Email: rshamsud@yahoo.com
-

Logging in Oracle

- Oracle maintains ACID properties of RDBMS
 - A- Atomicity
 - C- Consistency
 - I- Isolation
 - D-Durability

 - Oracle logging mechanism plays pivotal role in implementing few ACID properties
-

Logging concepts

- ❑ In theory, at least, three kinds of logging possible:
 - ❑ Physical – Each change will log the whole block
 - ❑ Logical – SQLs are logged
 - ❑ Physiological logging – Oracle uses this method
-

Logging in Oracle

□ Physiological logging

- An atomic change to a data block generates a change vector – Physical
 - Multiple change vectors are grouped together to generate redo records -Logical
-

Logging in Oracle

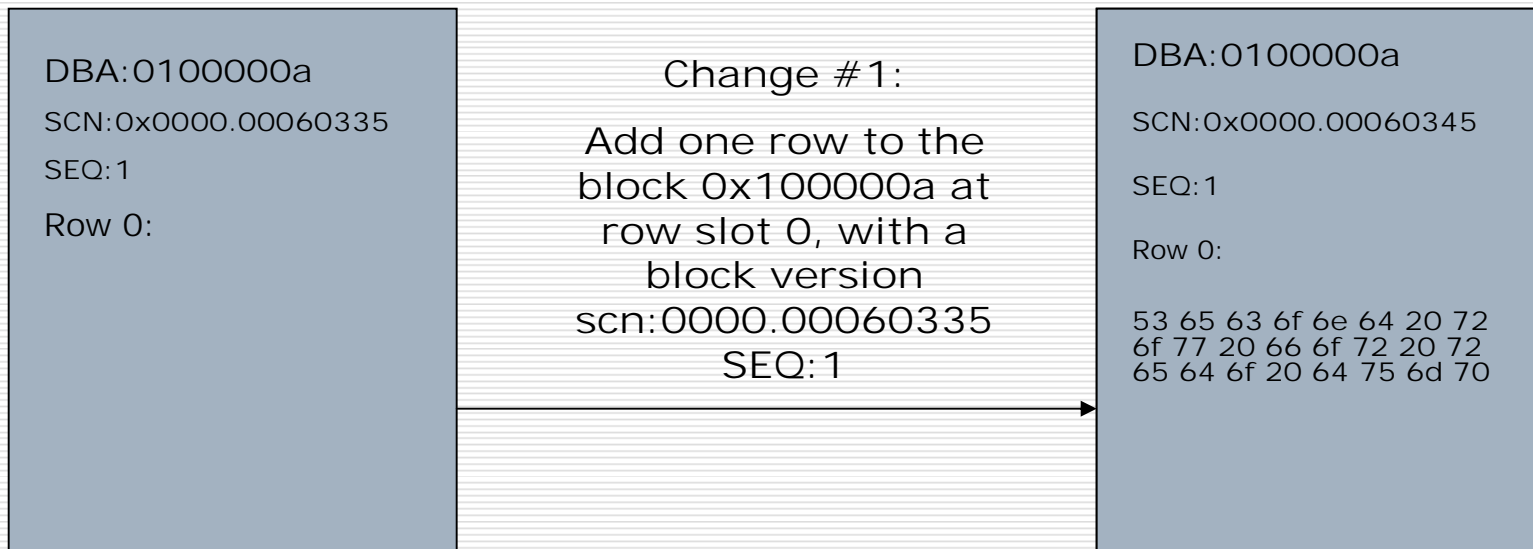
We will consider the following SQL:

```
insert into redo_dump  
values
```

```
  ('2', 'Second row for redo dump');
```

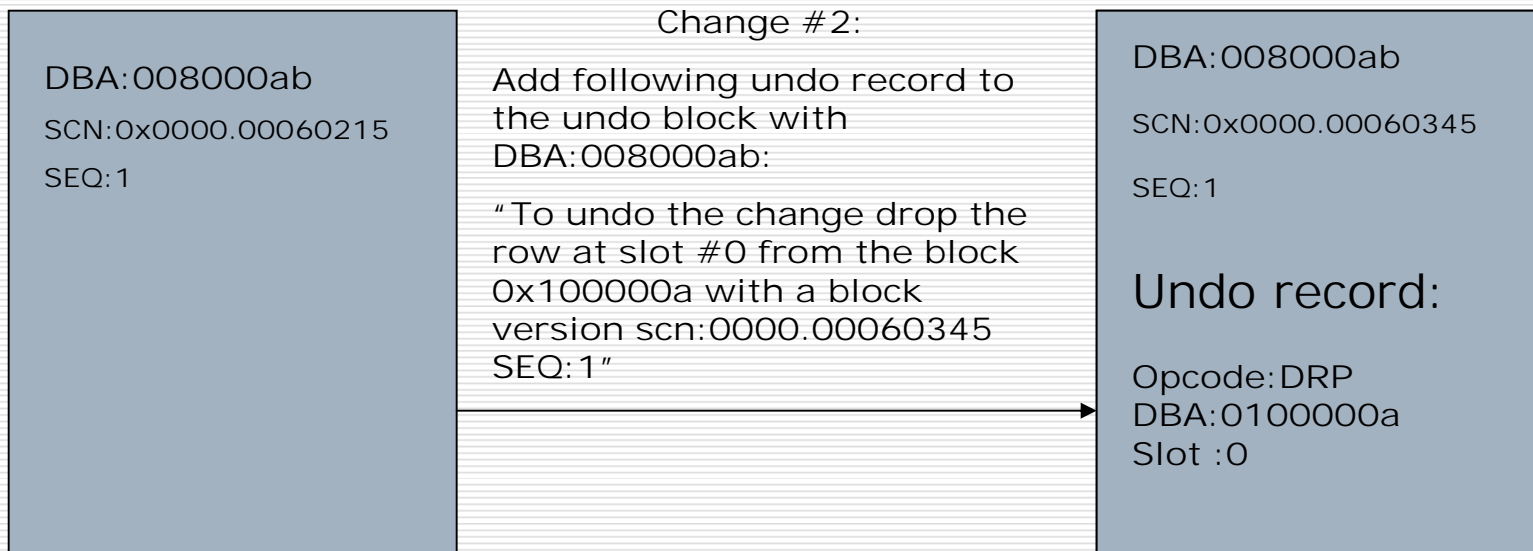
*This table has no indices..

Change vector (Simplified)



DBA:0100000a belongs to table segment

Undo change vector (Oversimplified)

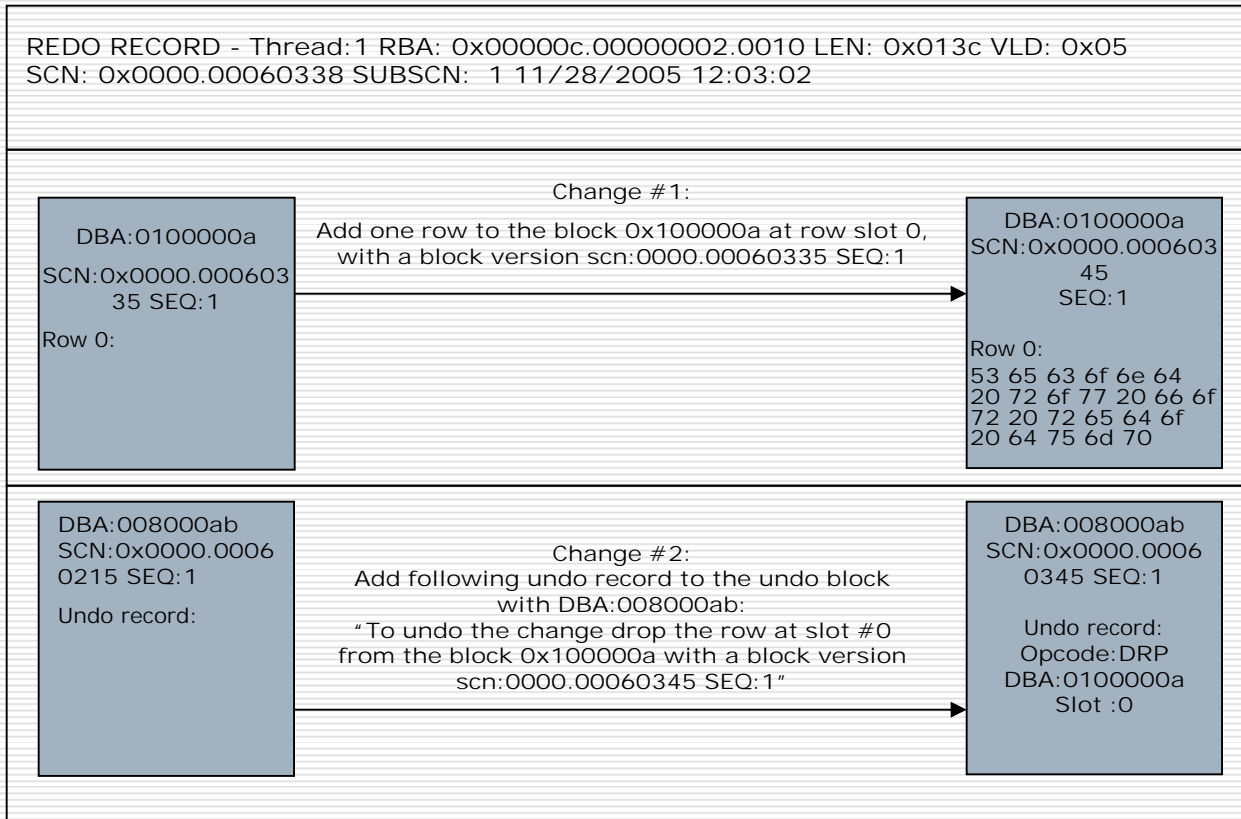


DBA:008000ab belongs to undo segment

Change vector

- ❑ A change vector transitions a database block from one version to another version.
 - ❑ A change vector also specifies the version of the block that this change vector can be applied to.
 - ❑ Database blocks has version information in its header, of the format <SCN,SEQ>.
 - ❑ One or more change vectors grouped together as redo record.
-

Redo record



insert into redo_dump
values (
'2',
'Second row for redo
dump');

Redo record

- ❑ Redo records transitions a database from one state to another state.
 - ❑ All or None of the change vectors will be applied from a redo record.
 - ❑ In a sense, it is a logical change.
-

Redo generation

- ❑ Processes intent to modify a database block
 - ❑ Creates redo records describing the change
 - ❑ Copies the redo records in to the log buffer
 - ❑ Then, these redo records are applied to the database blocks in the buffer cache, exactly as recovery.*

* Behavior is slightly different in 10g

Important rules..

Rule #1

Redo records must be generated before the data buffer is modified.

Important rules..

Rule #2

DBWR can write a dirty buffer only if LGWR has written the redo block, that includes the latest change to that block.

Important rules..

Rule #3

At commit time, LGWR confirms that redo records generated up to and including commit redo record is written to disk.

Redo ordering

- ❑ Ordering of the redo records maintained by $\langle \text{SCN}, \text{SUBSCN} \rangle$ pair.
 - ❑ Multiple redo records within the same SCN, may have different SUBSCN.
 - ❑ Multiple redo records at the same SCN, for a block will increment the sequence #.
Essentially block version is: $\langle \text{SCN}, \text{SEQ} \rangle$
-

Internals of redo: Inserts

Table structure:

```
create table redo_dump
(
char_column          char,
varchar2_column      varchar2(100)
);
```

SQL:

```
insert into redo_dump
values ('2', 'Second row for redo dump');
```

Internals of redo..

Redo record header

```
REDO RECORD - Thread: 1 RBA: 0x00000c.00000002.0010 LEN: 0x013c VLD: 0x05
SCN: 0x0000.00060338 SUBSCN: 1 11/28/2005 12: 03: 02

CHANGE #1 TYP: 0 CLS: 26 AFN: 2 DBA: 0x008000ab SCN: 0x0000.00060335 SEQ: 1 OP: 5.1
ktudb redo: slz: 68 spc: 6482 flg: 0x0022 seq: 0x0021 rec: 0x0f
            xid: 0x0005.011.00000020
ktubu redo: slt: 17 rcl: 14 opc: 11.1 objn: 9913 objd: 9913 tsn: 4
Undo type: Regular undo      Undo type: Last buffer split: No
Tablespace Undo: No
            0x00000000
KDO undo record:
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008000ab.0021.0e
KDO Op code: DRP row dependencies Disabled
            xtype: XA flags: 0x00000000 bdba: 0x0100000a hdba: 0x01000009
            ltl: 1 lspac: 0 maxfr: 4863
            tabn: 0 slot: 4(0x4)

CHANGE #2 TYP: 0 CLS: 1 AFN: 4 DBA: 0x0100000a SCN: 0x0000.00060335 SEQ: 1 OP: 11.2
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008000ab.0021.0f
KDO Op code: IRP row dependencies Disabled
            xtype: XA flags: 0x00000000 bdba: 0x0100000a hdba: 0x01000009
            ltl: 1 lspac: 0 maxfr: 4863
            tabn: 0 slot: 4(0x4) size/delt: 30
            fb: --H-FL-- lb: 0x1 cc: 2
            null: --
            col 0: [ 1] 32
            col 1: [24]
            53 65 63 6f 6e 64 20 72 6f 77 20 66 6f 72 20 72 65 64 6f 20 64 75 6d 70
```

Change vector for undo segment

Change vector for table segment

Redo record

Length: 316 bytes

Redo byte address:
Sequence#.block#.offset

REDO RECORD - Thread: 1 RBA: 0x00000c.00000002.0010
LEN: 0x013c VLD: 0x05

SCN: 0x0000.00060338 SUBSCN: 1 11/28/2005 12:03:02

CHANGE #1 TYP:0 CLS:26 AFN:2 DBA:0x008000ab SCN:0x0000.00060335 SEQ: 1 OP:5.1
ktudb redo: siz: 68 spc: 6482 flg: 0x0022 seq: 0x0021 rec: 0x0f
xid: 0x0005.011.00000020
ktubu redo: slt: 17 rci: 14 opc: 11.1 objn: 9913 objd: 9913 tsn: 4
Undo type: Regular undo Undo type: Last buffer split: No
itli: 1 ispac: 0 maxfr: 4863

tabn: 0 slot: 4(0x4)

CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0100000a SCN:0x0000.00060335 SEQ: 1 OP:11.2
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008000ab.0021.0f
tabn: 0 slot: 4(0x4) size/delt: 30

fb: --H-FL-- lb: 0x1 cc: 2
null: --
col 0: [1] 32
col 1: [24]
53 65 63 6f 6e 64 20 72 6f 77 20 66 6f 72 20 72 65 64 6f 20 64 75 6d 70

SCN at this RBA

5.1 is for
undo
block/hdr

Data block address
for undo block

Block version that
this vector can be
applied (scn, seq).

Change vector-undo block

REDO RECORD - Thread: 1 RBA: 0x00000c.00000002.0010 LEN: 0x013c WLD: 0x05
SCN: 0x0000.00060338 SUBSCN: 1 11/28/2005 12:03:02

Change
header

CHANGE #1 TYP: 0 CLS: 26 AFN: 2 DBA: 0x008000ab SCN: 0x0000.00060335
SEQ: 1 OP: 5.1

ktudb redo: siz: 68 spc: 6482 flg: 0x0022 seq: 0x0021 rec: 0x0f
xid: 0x0005.011.00000020

ktubu redo: slt: 17 rci: 14 opc: 11.1 obj n: 9913 obj d: 9913 tsn: 4

Undo type: Regular undo Undo type: Last buffer split: No

Tabl espace Undo: No

0x00000000

Transaction id
of this TX

KDO undo record:

KTB Redo

op: 0x02 ver: 0x01

op: C uba: 0x008000ab.0021.0e

KDO Op code: DRP row dependenci es Di sabl ed

xtype: XA flags: 0x00000000 bdba: 0x0100000a hdba: 0x01000009

itli: 1 ispac: 0 maxfr: 4863

tabn: 0 slot: 4(0x4)

CHANGE #2 TYP: 0 CLS: 1 AFN: 4 DBA: 0x0100000a SCN: 0x0000.00060335 SEQ: 1 OP: 11.2

KTB Redo

op: 0x02 ver: 0x01

....

col 0: [1] 32

col 1: [24]

53 65 63 6f 6e 64 20 72 6f 77 20 66 6f 72 20 72 65 64 6f 20 64 75 6d 70

Change vector-undo block

REDO RECORD - Thread: 1 RBA: 0x00000c.00000002.0010 LEN: 0x013c VLD: 0x05
 SCN: 0x0000.00060338 SUBSCN: 1 11/28/2005 12:03:02

CHANGE #1 TYP: 0 CLS: 26 AFN: 2 DBA: 0x008000ab SCN: 0x0000.00060335
 SEQ: 1 OP: 5.1

ktudb redo: siz: 68 spc: 6482 flg: 0x0022 seq: 0x0021 rec: 0x0f
 xid: 0x0005.011.00000020

ktubu redo: slt: 17 rci: 14 opc: 11.1 objn: 9913 objd: 9913 tsn: 4
 Undo type: Regular undo Undo type: Last buffer split: No

Tablespace Undo: No
 0x00000000

KDO undo record:

KTB Redo
 op: 0x02 ver: 0x01
 op: C uba: 0x008000ab.0021.0e
 KDO Op code: DRP row dependencies Disabled
 xtype: XA flags: 0x00000000 bdba: 0x0100000a hdba: 0x01000009
 itli: 1 ispac: 0 maxfr: 4863
 tabn: 0 slot: 4(0x4)

Undo byte address

Undo record

Undo for insert is drop row

Row in the row directory

Dbas of table block

Segment header

CHANGE #2 TYP: 0 CLS: 1 AFN: 4 DBA: 0x0100000a SCN: 0x0000.00060335 SEQ: 1 OP: 11.2

KTB Redo
 op: 0x02 ver: 0x01

col 0: [1] 32
 col 1: [24]

53 65 63 6f 6e 64 20 72 6f 77 20 66 6f 72 20 72 65 64 6f 20 64 75 6d 70

Change vector – table block

REDO RECORD - Thread:1 RBA: 0x00000c.00000002.0010 LEN: 0x013c VLD: 0x05
SCN: 0x0000.00060338 SUBSCN: 1 11/28/2005 12:03:02

CHANGE #1 TYP:0 CLS:26 AFN:2 DBA:0x008000ab SCN:0x0000.00060335 SEQ: 1 OP:5.1
ktudb redo: siz: 68 spc: 6482 flg: 0x0022 seq: 0x0021 rec: 0x0f
xid: 0x0005.011.00000020

.....
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 slot: 4(0x4)

CHANGE #2 TYP:0 CLS: 1 AFN: 4 DBA: 0x0100000a SCN: 0x0000.00060335 SEQ: 1
OP: 11.2

KTB Redo

op: 0x02 ver: 0x01

op: C uba: 0x008000ab.0021.0f

KDO Op code: IRP row dependencies Disabled

xtype: XA flags: 0x00000000 bdba: 0x0100000a hdba: 0x01000009

itli: 1 ispac: 0 maxfr: 4863

tabn: 0 slot: 4(0x4) size/delt: 30

fb: --H-FL-- lb: 0x1 cc: 2

null: --

col 0: [1] 32

col 1: [24]

53 65 63 6f 6e 64 20 72 6f 77 20 66 6f 72 20 72 65 64 6f 20 64 75 6d 70

DBA where
row resides

Block version
needed

Insert rowpiece: 11.2

Slot in the row directory

Flags for this row

Column values

Internals of redo: Update

```
insert into redo_internals_tbl values  
( 'A1', 'FIRST ROW' );
```

- Following SQL used to explain redo for single column single row update.

```
update redo_internals_tbl set  
varchar2_column = 'FIRST ROW UPD'  
where char_column = 'A1';
```

Redo record for 1 row/1 column update

REDO RECORD - Thread: 1 RBA: 0x0000e8.00000002.0010 LEN: 0x0140 VLD: 0x05
SCN: 0x0000.003f96f5 SUBSCN: 1 01/12/2006 11:46:07
CHANGE #1 TYP:0 CLS:24 AFN:2 DBA:0x008003b6 SCN:0x0000.003f96f1 SEQ: 2 OP:5.1
ktudb redo: siz: 100 spc: 3792 flg: 0x0022 seq: 0x03d6 rec: 0x28
xid: 0x0004.025.00001113

ktubu redo: slt: 37 rci: 39 opc: 11.1 objn: 14038 objd: 14038 tsn: 4
Undo type: Regular undo Undo type: Last buffer split: No

.....
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.27
KDO Op code: URP row dependencies Disabled
xtype: XA flags: 0x00000000 bdba: 0x0101868a hdba: 0x01018689
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 slot: 0(0x0) flag: 0x2c lock: 1 ckix: 1
ncol: 2 nnew: 1 size: -4

col 1: [9] 46 49 52 53 54 20 52 4f 57
CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0101868a SCN:0x0000.003f96f1 SEQ: 6 OP:11.5

KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.28
KDO Op code: URP row dependencies Disabled
xtype: XA flags: 0x00000000 bdba: 0x0101868a hdba: 0x01018689
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 slot: 0(0x0) flag: 0x2c lock: 1 ckix: 1
ncol: 2 nnew: 1 size: 4
col 1: [13] 46 49 52 53 54 20 52 4f 57 20 55 50 44

Column size to decrease by 4 bytes

Old values of row piece in undo vector :

F I R S T R O W

New values of the row piece :

F I R S T R O W U P D

Internals of redo: Delete

```
insert into redo_internals_tbl values  
('A1','FIRST ROW');
```

- Following delete statement used to explain the redo for delete.

```
delete from redo_internals_tbl  
where char_column = 'A1';
```

Redo record for 1 row delete

REDO RECORD - Thread: 1 RBA: 0x0000ea.00000002.0010 LEN: 0x0130 VLD: 0x05
SCN: 0x0000.003f97b3 SUBSCN: 1 01/12/2006 11:51:44
CHANGE #1 TYP:0 CLS:24 AFN:2 DBA:0x008003b6 SCN:0x0000.003f97af SEQ: 2 OP:5.1

ktudb redo: siz: 120 spc: 3376 flg: 0x0022 seq: 0x03d6 rec: 0x2c
xid: 0x0004.027.00001113
ktubu redo: slt: 39 rci: 43 opc: 11.1 objn: 14039 objd: 14039 tsn: 4
Undo type: Regular undo Undo type: Last buffer split: No

.....
KDO undo record:
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.2b
KDO Op code: IRP row dependencies Disabled
xtype: XA flags: 0x00000000 bdba: 0x0101898a hdba: 0x01018989
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 slot: 0(0x0) size/delt: 16
fb: --H-FL-- lb: 0x1 cc: 2
null:
col 0: [2] 41 31
col 1: [9] 46 49 52 53 54 20 52 4f 57

Undo for delete is
insert : IRP

Pre-image of the
row

11.3 for delete
row piece

CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0101898a SCN:0x0000.003f97af SEQ: 6 OP:11.3
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.2c
KDO Op code: DRP row dependencies Disabled
xtype: XA flags: 0x00000000 bdba: 0x0101898a hdba: 0x01018989
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 slot: 0(0x0)

Slot # in row
directory to delete

Internals : Insert with one index

Table structure:

```
create table redo_internals_tbl  
(  
  char_column      char(2),  
  varchar2_column  varchar2(20)  
);
```

Index structure:

```
create index redo_internals_idx on  
redo_internals_tbl (char_column);
```

Internals: Insert with one index

REDO RECORD - Thread: 1 RBA: 0x0000ee.00000002.0010 LEN: 0x0130 VLD: 0x05
SCN: 0x0000.003f9d6a SUBSCN: 1 01/12/2006 12:40:45
CHANGE #1 TYP:0 CLS:24 AFN:2 DBA:0x008003b6 SCN:0x0000.003f9d67 SEQ: 2 OP:5.1
.....
tabn: 0 slot: 1(0x1)
CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0100d60a SCN:0x0000.003f9d67 SEQ: 5 OP:11.2
.....
col 0: [2] 41 32
col 1: [10] 53 45 43 4f 4e 44 20 52 4f 57

Redo record
for table
block

REDO RECORD - Thread: 1 RBA: 0x0000ee.00000003.0010 LEN: 0x0108 VLD: 0x05
SCN: 0x0000.003f9d6b SUBSCN: 1 01/12/2006 12:40:45
CHANGE #1 TYP:0 CLS:24 AFN:2 DBA:0x008003b6 SCN:0x0000.003f9d6a SEQ: 1 OP:5.1
xid: 0x0004.02a.00001113
ktubu redo: slt: 42 rci: 53 opc: 10.22 objn: 14043 objd: 14043 tsn: 4
Undo type: Regular undo Undo type: Last buffer split: No
index undo for leaf key operations

Undo for leaf block change

...
Dump kdilk : itl=2, kdxlkflg=0x1 sdc=0 indexid=0x101b221 block=0x0101b222
(kdxlpu): purge leaf row
key : (10): 02 41 32 06 01 00 d6 0a 00 01
CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0101b222 SCN:0x0000.003f9d67 SEQ: 1 OP:10.2
index redo (kdxlin): insert leaf row
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.36
REDO: SINGLE / -- / --
itl: 2, sno: 1, row size 14
insert key: (10): 02 41 32 06 01 00 d6 0a 00 01

Layer 10 opcode 2

Index key entry

Multi row insert

```
insert into redo_internals_tbl
select
    substr(object_name, 1,2) char_column,
    substr(object_name, 1,10) varchar2_column
from dba_objects where rownum <4;
```

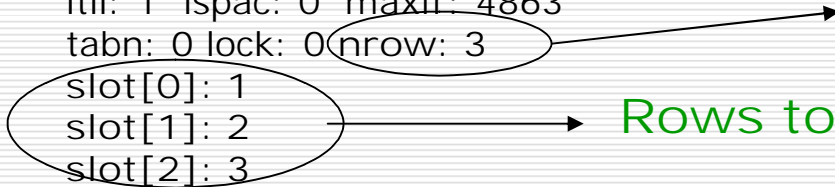
Internals: Multi row insert

REDO RECORD - Thread:1 RBA: 0x0000f0.00000002.0010 LEN: 0x0140 VLD: 0x05
SCN: 0x0000.003fa870 SUBSCN: 1 01/12/2006 14:34:44
CHANGE #1 TYP:0 CLS:24 AFN:2 DBA:0x008003b6 SCN:0x0000.003fa86d SEQ: 1 OP:5.1
ktudb redo: siz: 76 spc: 2140 flg: 0x0022 seq: 0x03d6 rec: 0x38
 xid: 0x0004.02b.00001113
ktubu redo: slt: 43 rci: 55 opc: 11.1 objn: 14044 objd: 14044 tsn: 4
Undo type: Regular undo Undo type: Last buffer split: No

.....
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x008003b6.03d6.37
KDO Op code: QMD row dependencies Disabled
 xtype: XA flags: 0x00000000 bdba: 0x0101110a hdba: 0x01011109
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 lock: 0 nrow: 3
slot[0]: 1
slot[1]: 2
slot[2]: 3

3 rows to undo

Rows to delete for undo



Internals: Multi row insert

CHANGE #2 TYP:0 CLS: 1 AFN:4 DBA:0x0101110a SCN:0x0000.003fa86d SEQ: 5

OP:11.11

Multi row insert
opcode

KTB Redo

op: 0x02 ver: 0x01

op: C uba: 0x008003b6.03d6.38

KDO Op code: QMI row dependencies Disabled

Quick Multi row
insert

xtype: XA flags: 0x00000000 bdba: 0x0101110a hdba: 0x01011109

itli: 1 ispac: 0 maxfr: 4863

tabn: 0 lock: 1 nrow: 3

slot[0]: 1

tl: 11 fb: --H-FL-- lb: 0x0 cc: 2

col 0: [2] 43 4f

col 1: [4] 43 4f 4e 24

slot[1]: 2

tl: 13 fb: --H-FL-- lb: 0x0 cc: 2

col 0: [2] 49 5f

col 1: [6] 49 5f 43 4f 4c 32

slot[2]: 3

tl: 14 fb: --H-FL-- lb: 0x0 cc: 2

col 0: [2] 49 5f

col 1: [7] 49 5f 55 53 45 52 23

Changes for table
segment block to insert 3
rows

Global temporary tables

```
create global temporary table redo_internals_tbl  
(  
    char_column char(2),  
    Varchar2_column varchar2(20)  
) on commit delete rows  
;
```

```
insert into redo_internals_tbl  
select substr(object_name,1,2) char_columns,  
substr(object_name,1,20) varchar2_column  
from dba_objects where rownum < 11;
```

Global temporary tables

```
REDO RECORD - Thread: 1 RBA: 0x0001ae.00000002.0010 LEN: 0x00d0 VLD: 0x05
SCN: 0x0000.0057b223 SUBSCN: 1 02/03/2006 13:40:24
CHANGE #1 TYP:0 CLS:18 AFN:2 DBA:0x0080022c SCN:0x0000.0057b220 SEQ: 1 OP:5.1
ktudb redo: siz: 92 spc: 7238 flg: 0x0022 seq: 0x056d rec: 0x09
          xid: 0x0001.02d.0000109d
ktubu redo: slt: 45 rci: 8 opc: 11.1 objn: 17036 objd: 4199433 tsn: 3
Undo type: Regular undo      Undo type: Last buffer split: No
Tablespace Undo: No
          0x00000000
KDO undo record:
KTB Redo
op: 0x02 ver: 0x01
op: C uba: 0x0080022c.056d.08
KDO Op code: QMD row dependencies Disabled
  xtype: XA flags: 0x00000000 bdba: 0x0040140a hdba: 0x00401409
itli: 1 ispac: 0 maxfr: 4863
tabn: 0 lock: 0 nrow: 10
slot[0]: 1
slot[1]: 2
slot[2]: 3
slot[3]: 4
slot[4]: 5
slot[5]: 6
slot[6]: 7
slot[7]: 8
slot[8]: 9
slot[9]: 10
```

Redo generated for undo
segment only. No redo
for table segment blocks

Internals: nologging

- ❑ Nologging operations generate block range invalidate redo.
- ❑ Series of blocks invalidated.
- ❑ Recovery will mark these blocks as invalid.

```
insert /* + append */ into redo_internals_tbl
select substr(object_name,1,2),
       substr(object_name,1,10)
from dba_objects;
```

Internals: nologging

REDO RECORD - Thread:1 RBA: 0x0001a4.00000003.0054 LEN:
0x0034 VLD: 0x01

SCN: 0x0000.00564f6f SUBSCN: 1 02/01/2006 15:48:19

CHANGE #1 INVLD AFN:4 DBA:0x0100a213 BLKS:0x0019

SCN:0x0000.00564f6f SEQ: 1 OP:19.2

Direct Loader invalidate block range redo entry



Starting with block 0x0100a213,
25 blocks marked invalid. Insert
statement formatted blocks and wrote
them directly with minimal redo.

LOBs

- ❑ If the lob is stored inline, then redo/undo will be identical to normal rows.
 - ❑ Lob stored out-of-line, occupies at least one block per column value.
-

LOBs

- ❑ Out of line LOBs can be created with either logging or nologging attributes.
 - ❑ Logging – dumps the whole block to redo
 - ❑ Nologging – only block invalidation redo is generated for that column change.
-

LOBs

❑ Nologging

REDO RECORD - Thread:1 RBA: 0x0001b6.00000003.0138 LEN: 0x0034 VLD: 0x01
SCN: 0x0000.0057b6a0 SUBSCN: 1 02/03/2006 13:59:04
CHANGE #1 INVL D AFN:4 DBA:0x010014f5 BLKS:0x0001 SCN:0x0000.0057b6a0 SEQ: 1 OP:19.2
Direct Loader invalidate block range redo entry

52 bytes

❑ Logging

REDO RECORD - Thread:1 RBA: 0x0001b8.00000004.0138 LEN: 0x2024 VLD: 0x01
SCN: 0x0000.0057bf05 SUBSCN: 1 02/03/2006 15:26:24
CHANGE #1 TYP:1 CLS: 1 AFN:4 DBA:0x0100152d SCN:0x0000.0057bf05 SEQ: 1 OP:19.1
Direct Loader block redo entry

Long field block dump:

Object Id 17056

LobId: 00010000A514 PageNo 0

Version: 0x0000.00000000 pdba: 0

43 4f 4e 24 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20
20 20

....

8228 bytes

Internals: BeginTrans

- First DML change starts a transaction.

 - A slot is assigned in the transaction table for this transaction.

 - A transaction id is of the form:
 - Undo segment #. Slot#.Sequence#
-

Internals: BeginTrans

Undo segment header

REDO RECORD - Thread:1 RBA: 0x00019c.00000002.0010 LEN: 0x019c VLD: 0x0d
SCN: 0x0000.00564a9c SUBSCN: 2 02/01/2006 15:09:39
CHANGE #1 TYP:2 CLS: 1 AFN:4 DBA:0x0101c992 SCN:0x0000.005644f7 SEQ: 1 OP:11.2

....
col 1: [9] 46 49 52 53 54 20 52 4f 57

CHANGE #2 TYP:0 CLS:23 AFN:2 DBA:0x00800039 SCN:0x0000.00564a91 SEQ: 2 OP:5.2
ktudh redo: slt: 0x0007 sqn: 0x00001566 flg: 0x0012 siz: 108 fbi: 0
uba: 0x00800f94.04b9.4c pxid: 0x0000.000.00000000

CHANGE #3 TYP:0 CLS:24 AFN:2 DBA:0x00800f94 SCN:0x0000.00564a91 SEQ: 1 OP:5.1
ktudb redo: siz: 108 spc: 1048 flg: 0x0012 seq: 0x04b9 rec: 0x4c
xid: 0x0004.007.00001566

ktubl redo: slt: 7 rci: 0 opc: 11.1 objn: 16394 objd: 16394 tsn: 4
Undo type: Regular undo Begin trans Last buffer split: No

....
prev ctl max cmt scn: 0x0000.00563167 prev tx cmt scn: 0x0000.00563169
txn start scn: 0x0000.00564a7a logon user: 26 prev brb: 8392587 prev bcl: 0 KDO undo
record:

....
tabn: 0 slot: 1(0x1)

Xid: undo.slot.sqn

Undo segment header updates : New slot
allocated for this transaction in the
transaction table

Slot# x07 with sqn# x001566

Internals: Commit

Commit generates redo records marking commits:

Change vectors for commit can be piggybacked with other change vectors.

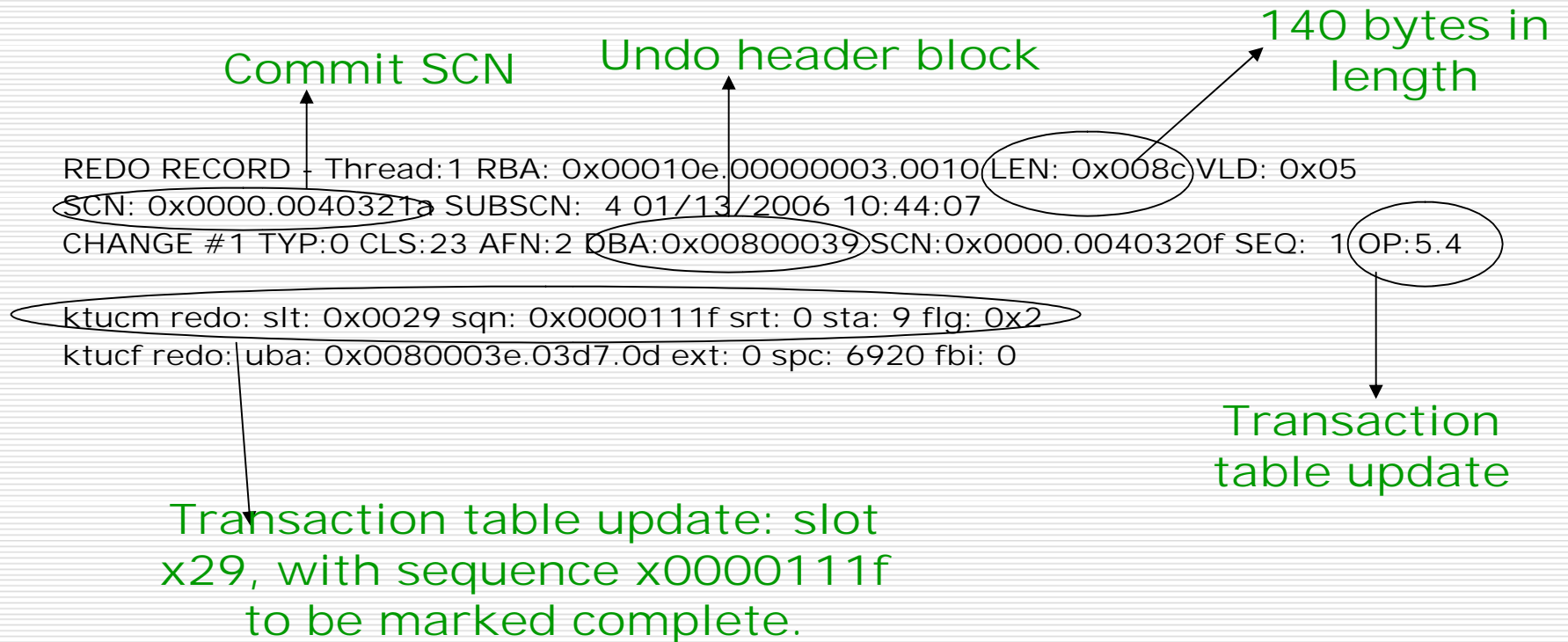
(or)

Can create a separate redo record just for the commit.

In 10g:

- separate redo record size is 140 bytes
 - Overhead when the commit is piggybacked is 72 bytes.
-

Internals: Commit



Internals: Rollback

Rollback of a transaction follows the undo link and rolls back the changes applying undo records!

```
insert into redo_internals_tbl values  
('A2','SECOND ROW');  
rollback;
```

Internals: Rollback

REDO RECORD - Thread:1 RBA: 0x0001aa.00000002.0010 LEN: 0x019c VLD: 0x0d
SCN: 0x0000.00572f87 SUBSCN: 1 02/02/2006 13:55:03
CHANGE #1 TYP:2 CLS: 1 AFN:4 DBA:0x010010fa SCN:0x0000.00572f84 SEQ: 5 OP:11.2

....
op: Z
KDO Op code: DRP row dependencies Disabled
 xtype: XA flags: 0x00000000 bdba: 0x010010fa hdba: 0x010010f9
itli: 2 ispac: 0 maxfr: 4863
tabn: 0 slot: 1(0x1)

Undo record for
insert statement

REDO RECORD - Thread:1 RBA: 0x0001aa.00000003.0010 LEN: 0x00c0 VLD: 0x05
SCN: 0x0000.00572f88 SUBSCN: 1 02/02/2006 13:55:03
CHANGE #1 TYP:0 CLS: 1 AFN:4 DBA:0x010010fa SCN:0x0000.00572f87 SEQ: 1 OP:11.3
KTB Redo

Rollback
applying
the undo
record

op: 0x03 ver: 0x01
op: Z
KDO Op code: DRP row dependencies Disabled
 xtype: XR flags: 0x00000000 bdba: 0x010010fa hdba: 0x010010f9
itli: 2 ispac: 0 maxfr: 4863
tabn: 0 slot: 1(0x1)
CHANGE #2 TYP:0 CLS:23 AFN:2 DBA:0x00800039 SCN:0x0000.00572f87 SEQ: 1 OP:5.11
ktubu redo: slt: 16 rci: 0 opc: 11.1 objn: 16936 objd: 16936 tsn: 4
Undo type: Regular undo Undo type: User undo done Begin trans Last buffer split: No
Tablespace Undo: No
 0x00000000

Transaction
table update.
Marks the
transaction as
rolled back

REDO RECORD - Thread:1 RBA: 0x0001aa.00000003.00d0 LEN: 0x0050 VLD: 0x01
SCN: 0x0000.00572f89 SUBSCN: 1 02/02/2006 13:55:03
CHANGE #1 TYP:0 CLS:23 AFN:2 DBA:0x00800039 SCN:0x0000.00572f88 SEQ: 1 OP:5.4
ktucm redo: slt: 0x0010 sqn: 0x00001573 srt: 0 sta: 9 flg: 0x4
rolled back transaction

Detecting redo size

- ❑ Three methods available:
 - ❑ Using v\$log_history
 - ❑ Using AWR
 - ❑ Using archivelog backup size
-

V\$log_history

@log_hist_daily_size.sql

Date DayOfWeek Redo size

14-JAN-06	SATURDAY	****	(86)
15-JAN-06	SUNDAY	*****	(129)
16-JAN-06	MONDAY	*****	(134)
17-JAN-06	TUESDAY	*****	(163)
18-JAN-06	WEDNESDAY	*****	(175)
19-JAN-06	THURSDAY	*****	(165)
20-JAN-06	FRI DAY	*****	(148)
21-JAN-06	SATURDAY	*****	(115)
22-JAN-06	SUNDAY	*****	(117)
23-JAN-06	MONDAY	*****	(101)

Using AWR

@redo_size_awr.sql

DB_NAME	REDO_DATE	redo_size (GB)
AFTP	14-JAN-06	78.04
AFTP	15-JAN-06	123.09
AFTP	16-JAN-06	127.88
AFTP	17-JAN-06	154.77
AFTP	18-JAN-06	161.79
AFTP	19-JAN-06	155.56
AFTP	20-JAN-06	145.79
AFTP	21-JAN-06	101.52
AFTP	22-JAN-06	112.68
AFTP	23-JAN-06	95.89

Breakdown of redo

- ❑ Logminer can be used to breakdown the redo size, at segment level.
 - ❑ V\$logmnr_contents has RBA and that can be used to find the redo size for the current change.
-

Breakdown of redo

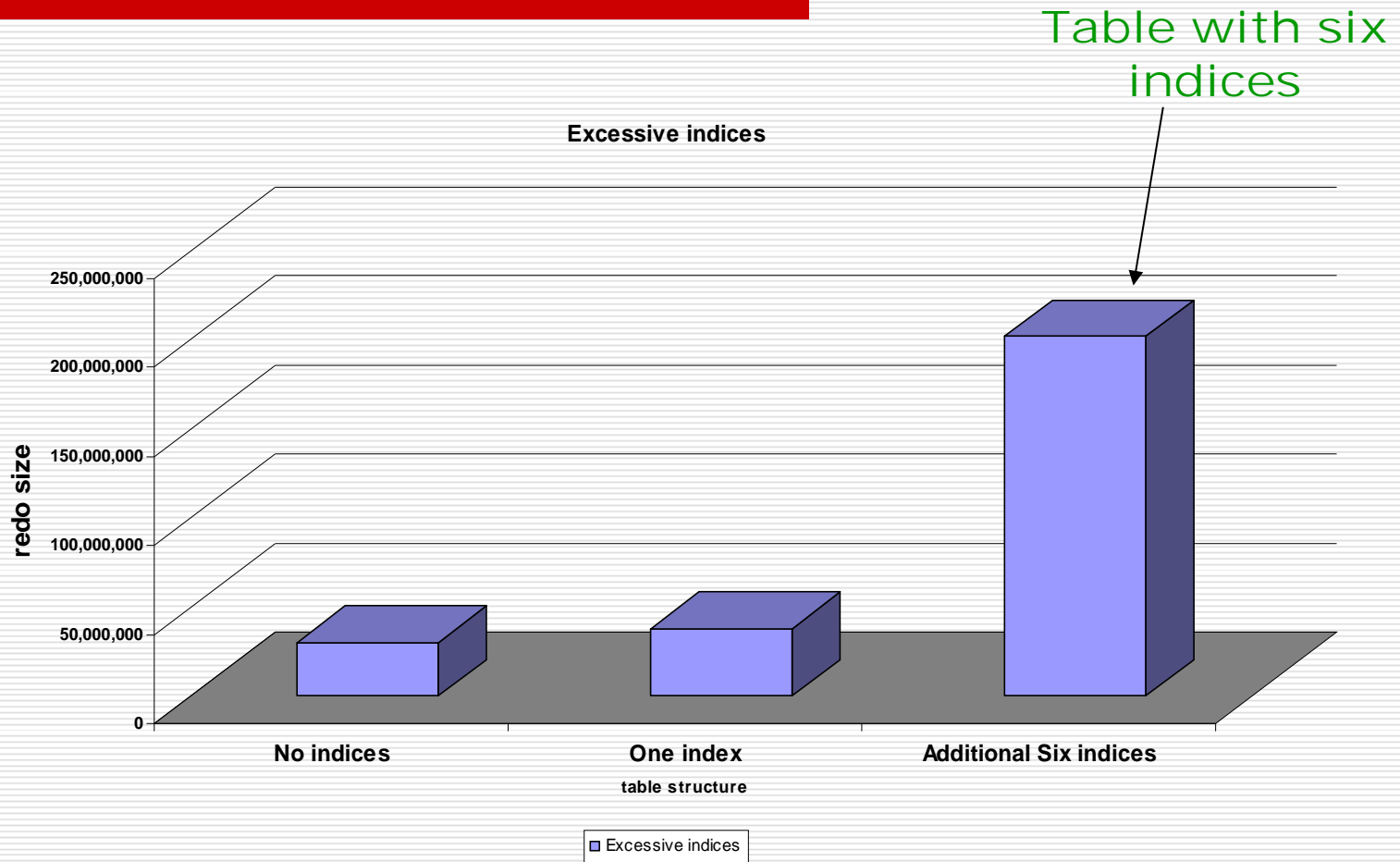
DATA_OBJ#	OPER	OBJ_NAME	TOTAL_REDO
321840	DELETE	FORECAST_BOC_EOC_UNITS_TBL	87781828
971566	UNSUPPORTED	BASE1\$ILP_PK	89723495
426226	INTERNAL	TMP_MERCHLOC_IDX	105794014
321842	INTERNAL	FORECAST_BOC_EOC_PK	109387686
971316	UNSUPPORTED	BASE1\$ILP_PK	112314947
71266	UPDATE	ALLOC_RESULTS_TBL	115642396
99814	INTERNAL	ALLOC_NEED_RESULTS_PK	127971405
226985	DELETE	TMP_FORECAST_UNITS_TBL	137166430
100071	DELETE	BOC_EOC_COUNTS_TBL	159137838
	0 COMMIT	LAST_WEEK_TRK_TBL	167614243
	0 COMMIT	PLNT_ITEMS_PROCESSED_TBL	167614243
	0 COMMIT	PLNU_ITEMS_PROCESSED_TBL	167614243
	0 COMMIT	PLNU_FAILED_ITEMS_TBL	167614243
	0 COMMIT	internal	167695464
71503	INTERNAL	BASE2\$ELIG_PLAN_PK	174174023
100072	INTERNAL	BOC_EOC_COMP_IDX	179380273
72637	INTERNAL	BASE2\$ELIG_MERCH_ID	261079376
321839	UPDATE	FORECAST_MERCH_LOCATIONS_TBL	368355146

Common causes of excessive redo

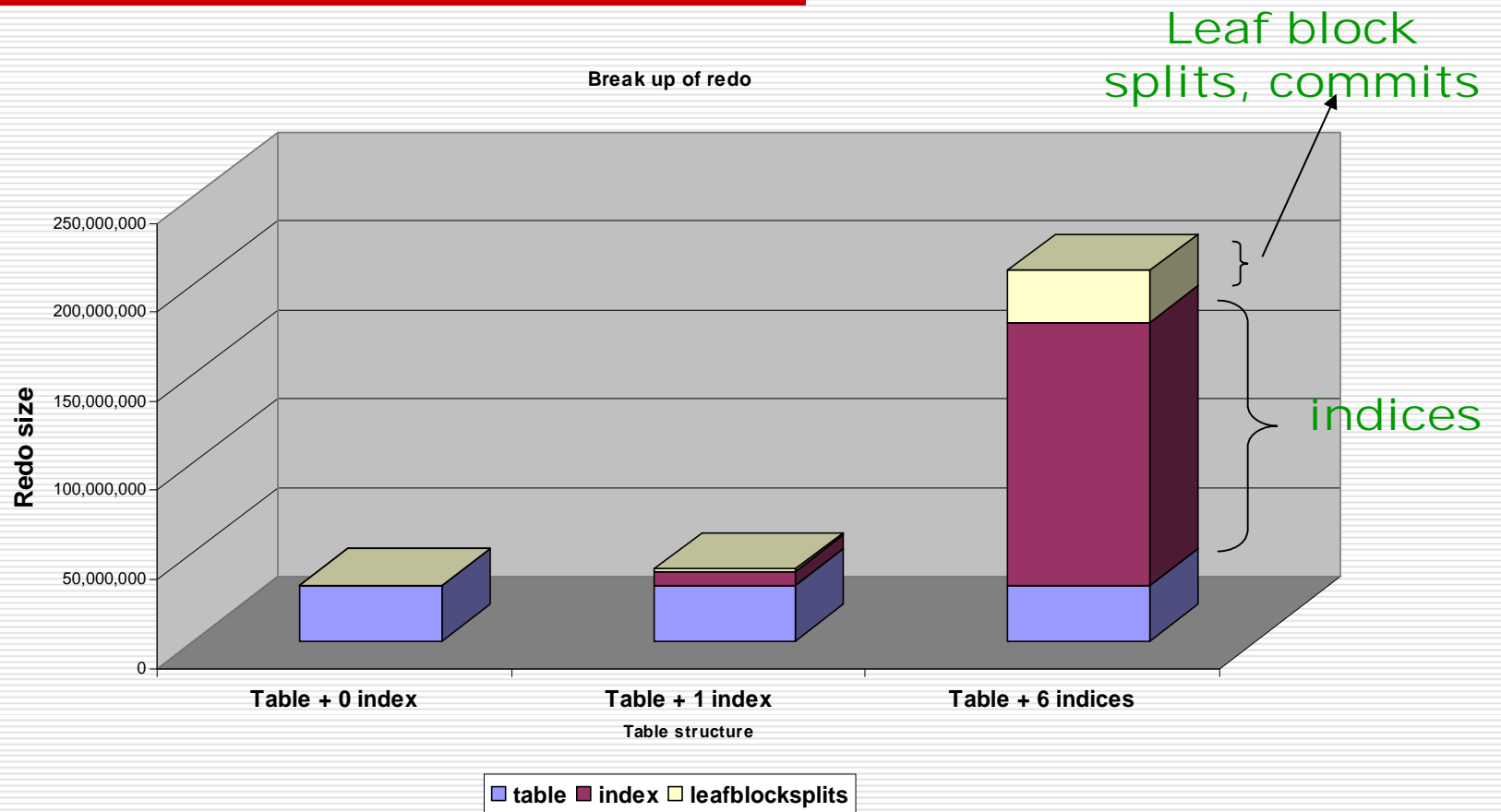
Excessive indices

- ❑ DML (Ins,upd,del,merge) generates redo for the index updates.
 - ❑ Too many indices can increase the redo size.
 - ❑ This test case compares table with no index, one index and six indices
-

Excessive indices



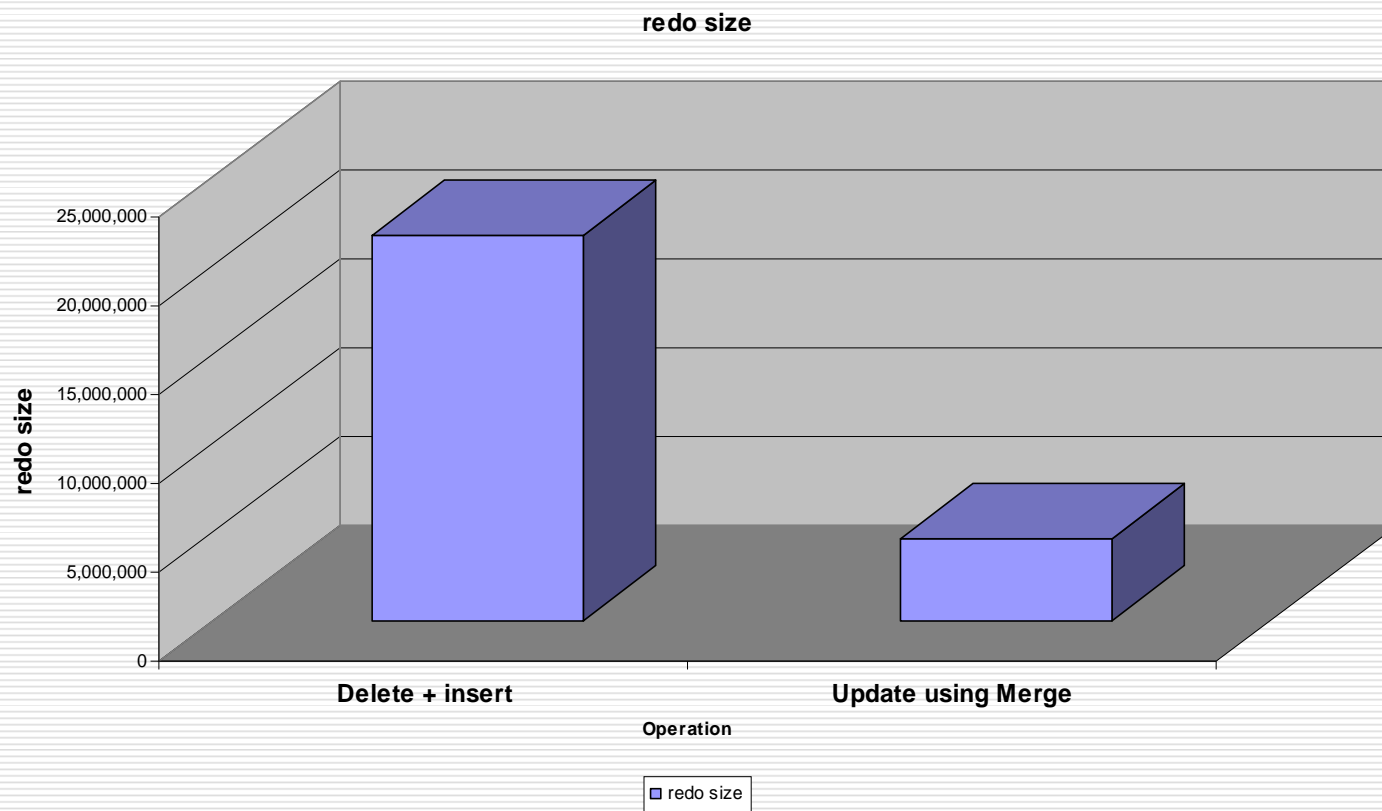
Redo size breakup



Del+ins vs Merge

- Typical coding practice is:
 - Delete from table1 where key1 = :1;
 - Insert into table1 values (...);
 - Insert into table1 values (...);
 - Costlier due to redo size. Option is to use updates or merge statements.
 - Merge into table1
 - .. When matched then
 - update..
 - when not matched then
 - insert..
-

Del+ins vs Merge



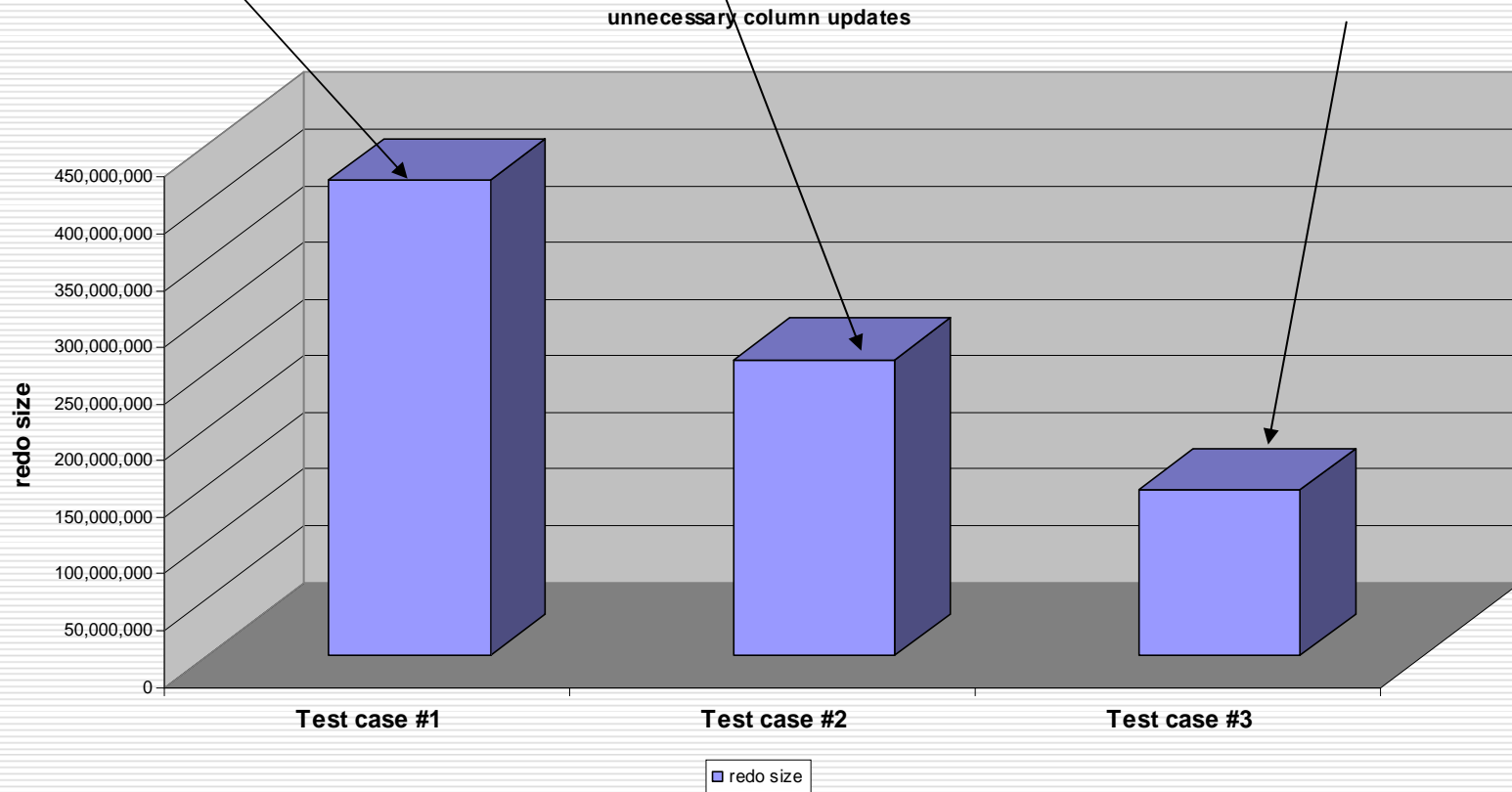
Unnecessary column updates

- Many tools update the columns even if there is no value change!
 - Oracle does not compare old and new values, before updates..
 - These unnecessary updates generates redo and some cases excessively...
-

All column updates

Updates to 3 varchar2(100) & 3 number columns

Updates to 1 varchar2(100) column and 3 number column



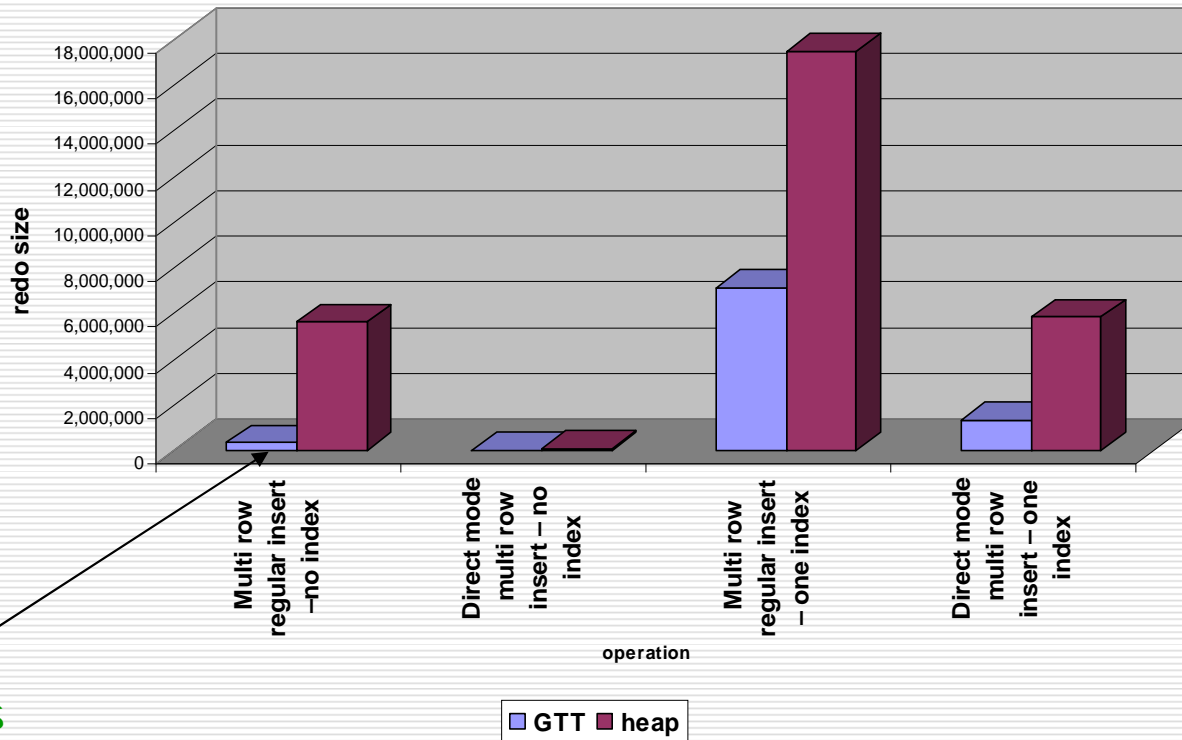
Redo for Global Temporary Tables

- GTTs are an option if the need is to keep the data temporarily in the session.

 - Use of GTT, reduces redo markedly
 - Redo generated only for undo blocks.
 - No redo generated for GTT table blocks, as they are in temp tablespace.
-

Redo for GTTs

redo size for GTT

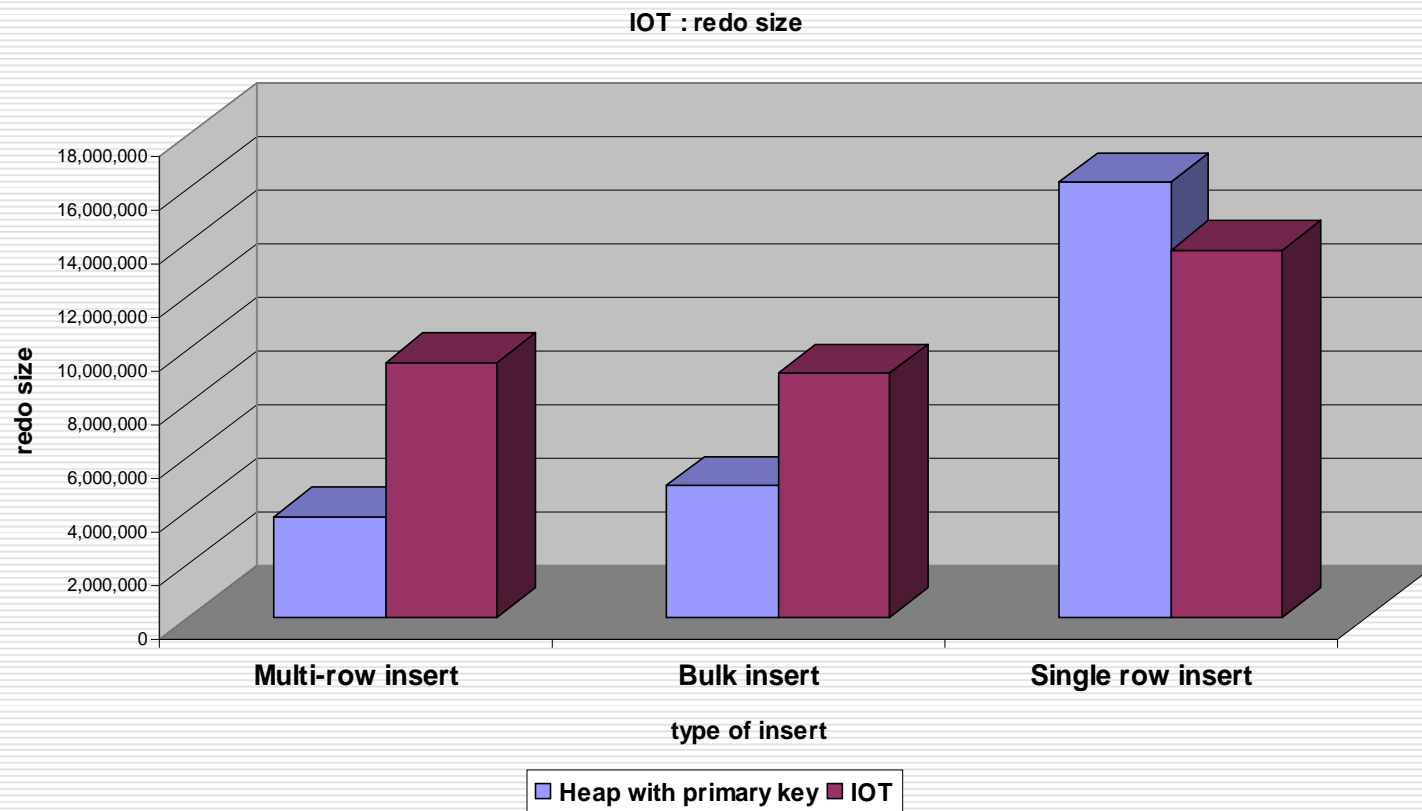


Redo generated only for undo blocks

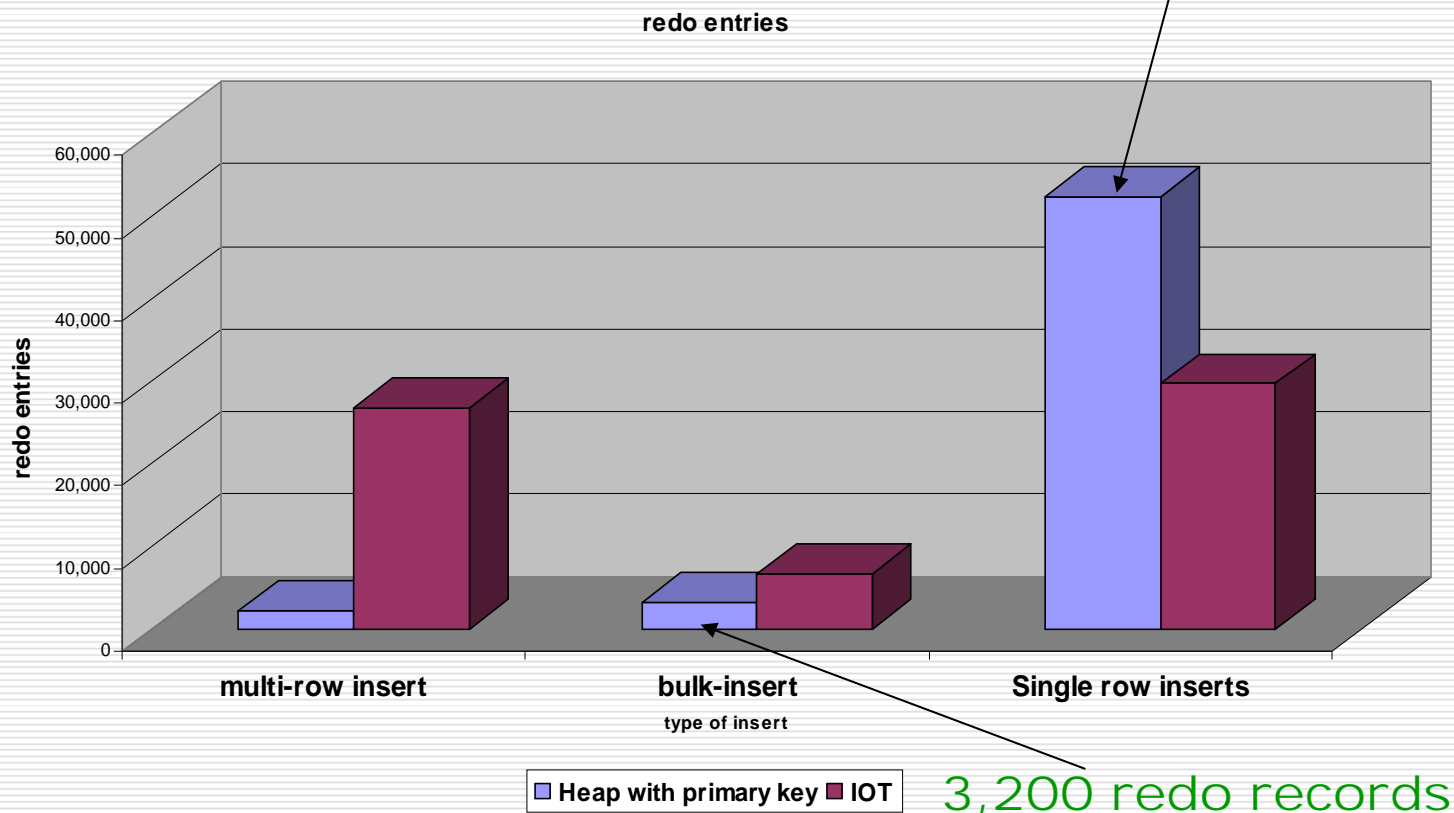
Use of IOT

- ❑ Index organized tables can be used to reduce redo in few scenarios.
 - ❑ Candidates to consider:
 - Tables with many indices with leading primary columns
 - Fewer # of columns
 - Access to the table is always through primary key etc.
 - ❑ Careful testing needed though
-

IOT : redo size



IOT:redo entries



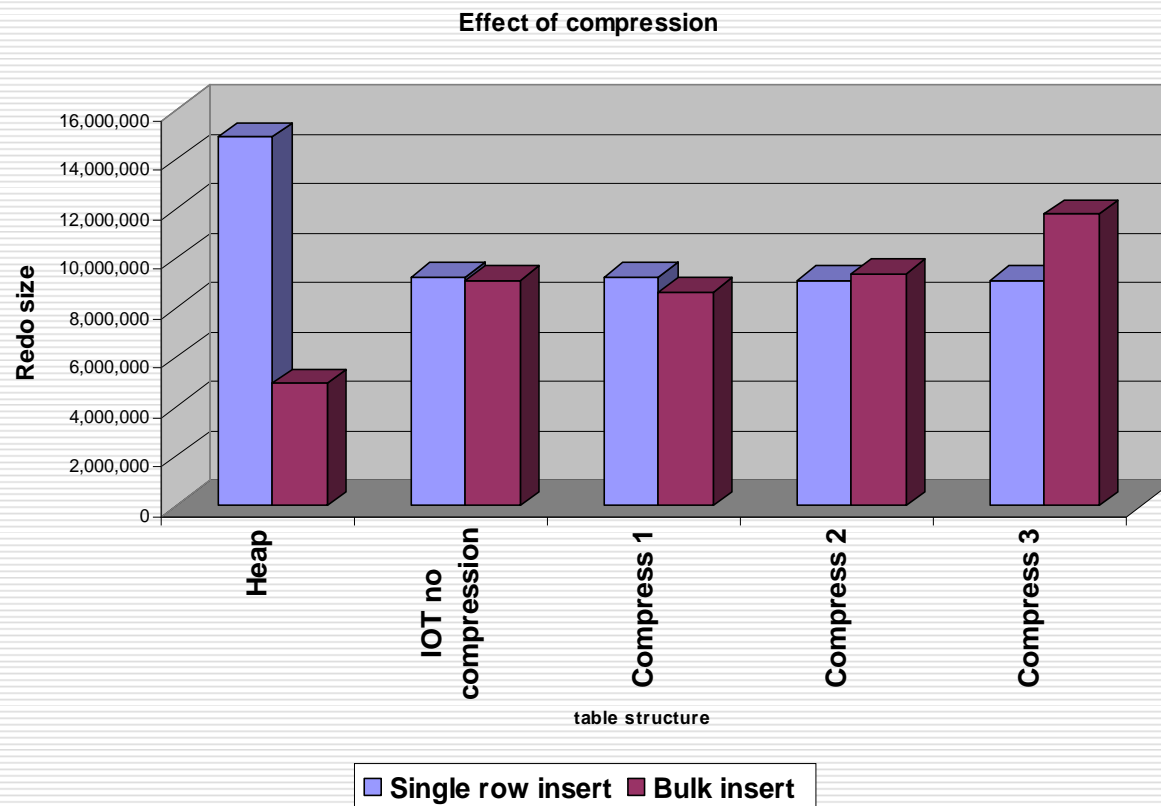
52,000 redo records
for 26,000 rows

3,200 redo records
for 26,000 rows

Compressed IOT

- ❑ Repeating values within a blocks are stored only once!
 - ❑ Depending upon data properties, space savings can be achieved.
 - ❑ Redo size depends upon data properties too!
-

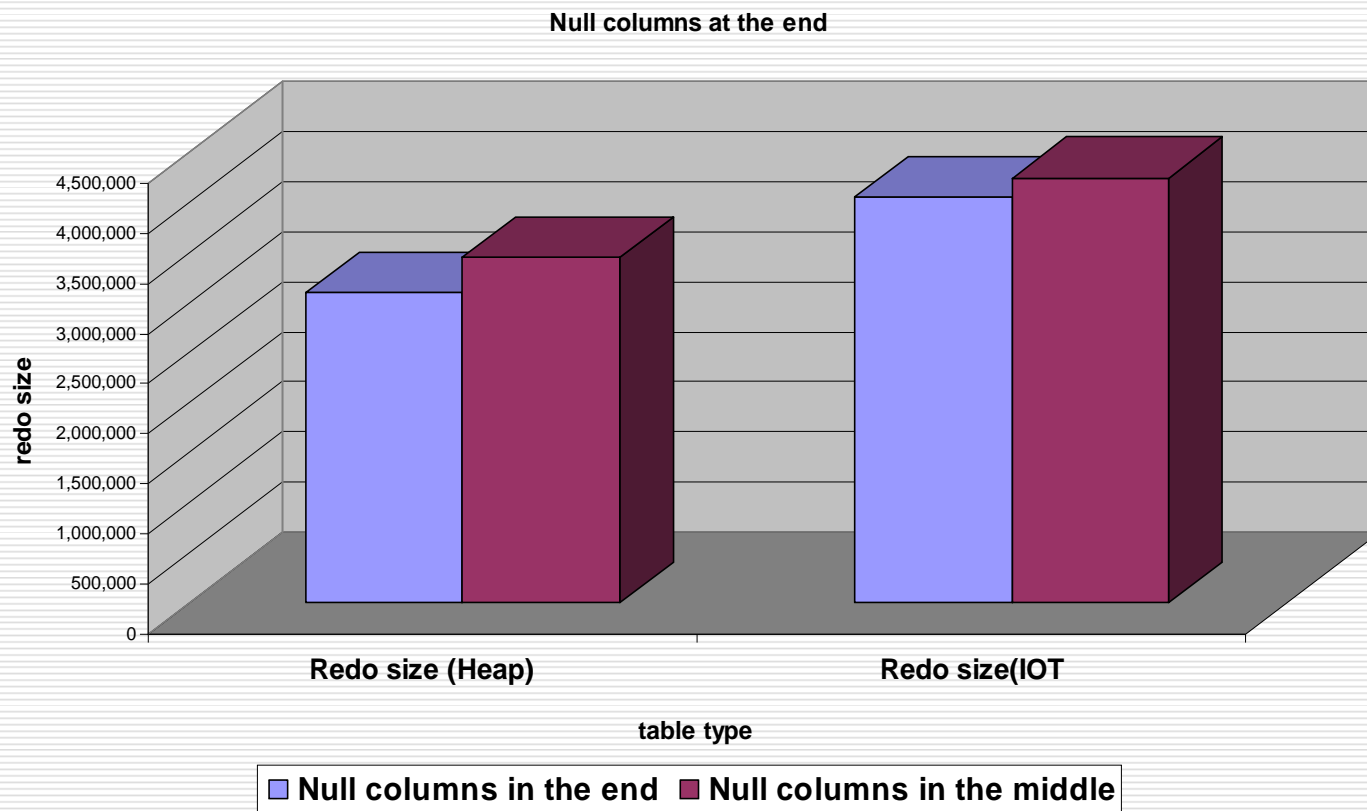
Compressed IOT



Null columns at the end

- ❑ Columns with null values stored explicitly, only if there is a column with non-null value, after that column.
 - ❑ Benefit depends upon # of columns and data distribution.
-

Null columns at the end



Reduction in scale

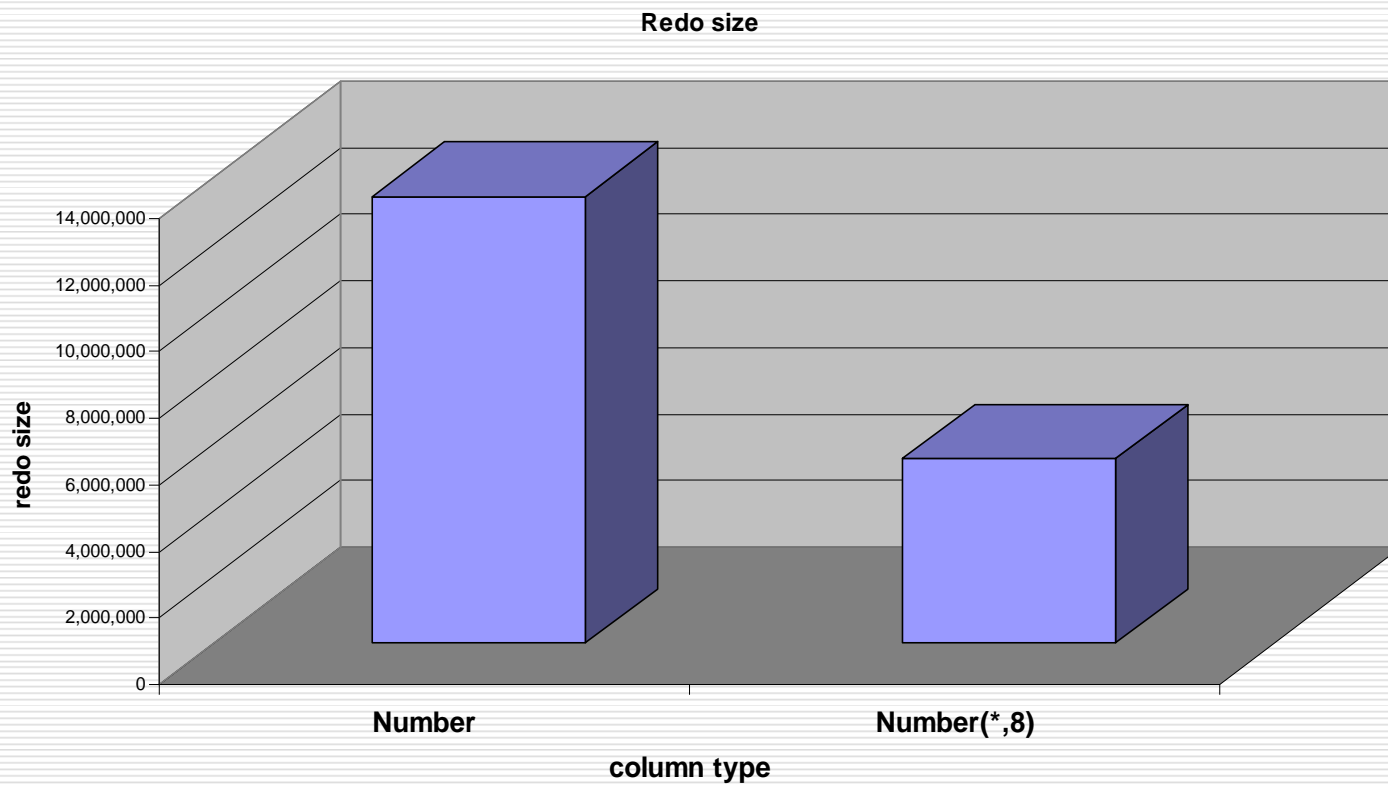
- Arithmetic operation populating directly in to a number column, defined without scale, can increase the length of that column, with a side effect of redo size increase.

For e.g.

```
create table test_arit (a number, b number(*,8));
insert into test_arit values (100/3.14, 100/3.14);
select length(a) , length(b) from test_arit;
```

LENGTH(A)	LENGTH(B)
-----	-----
39	11

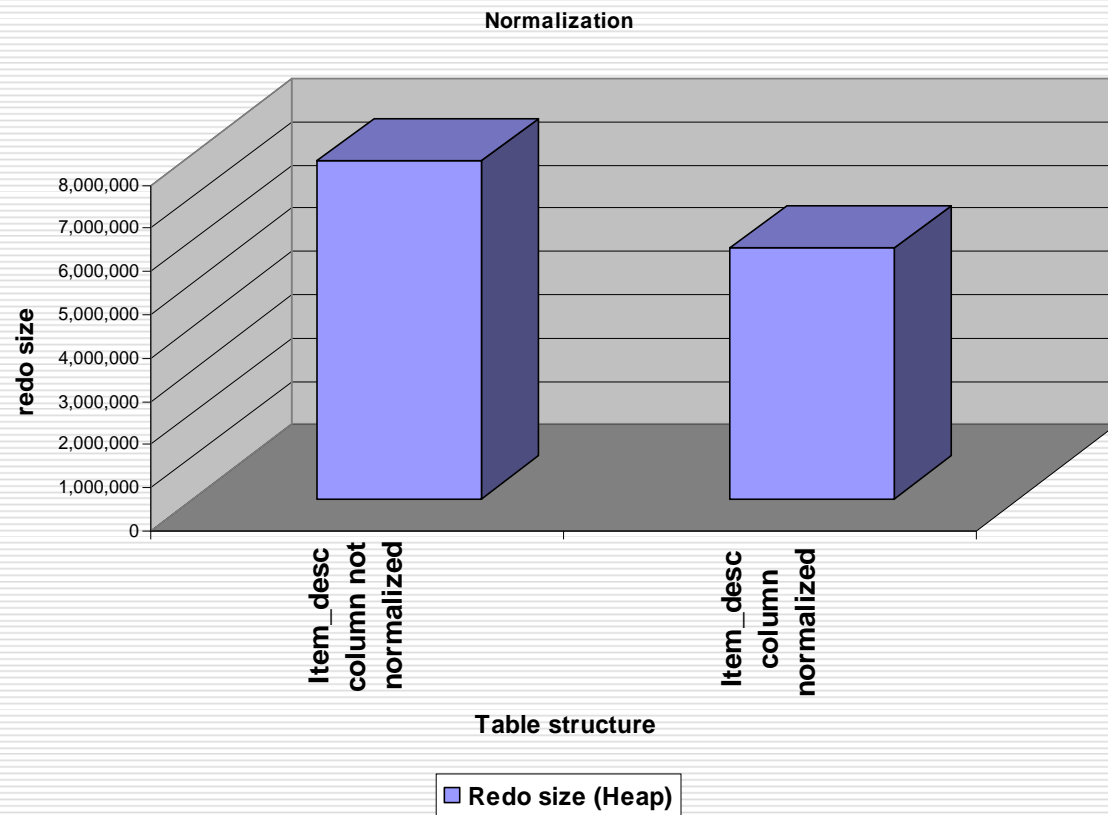
Reduction in scale



Normalize tables

- ❑ For very large tables, it pays to normalize them.
 - ❑ Keeping columns in the table itself, can increase segment size and redo size.
-

Normalize tables



Nologging inserts

- ❑ Nologging inserts generates minimal redo.

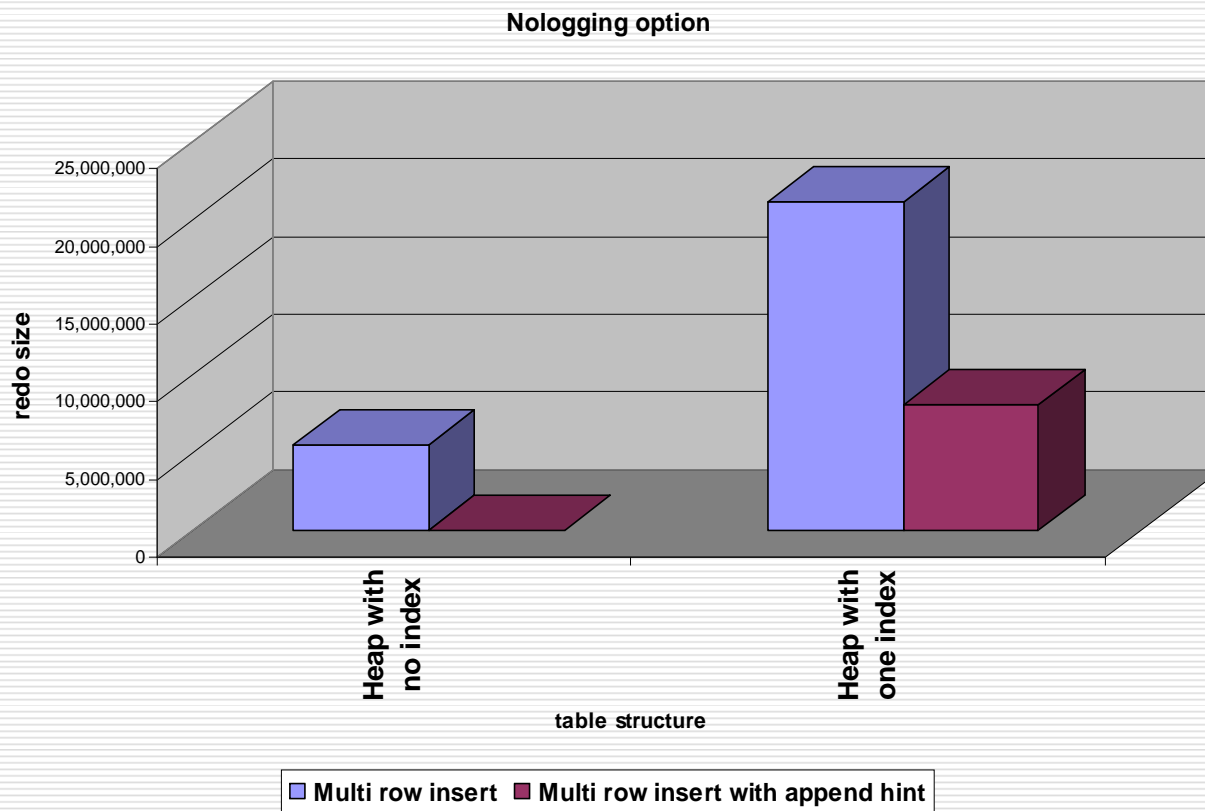
Insert /*+ append */

- ❑ Range of blocks invalidated with extent invalidation redo. Recovery may be affected.
 - ❑ Rows pre-formatted as blocks and written directly
-

Nologging inserts

- ❑ Still, redo generated for indices.
 - ❑ IOT is an index structure, so redo generated even if append hint specified.
 - ❑ Works only for multi-row insert.
 - ❑ Doesn't work for bulk insert or single row inserts.
-

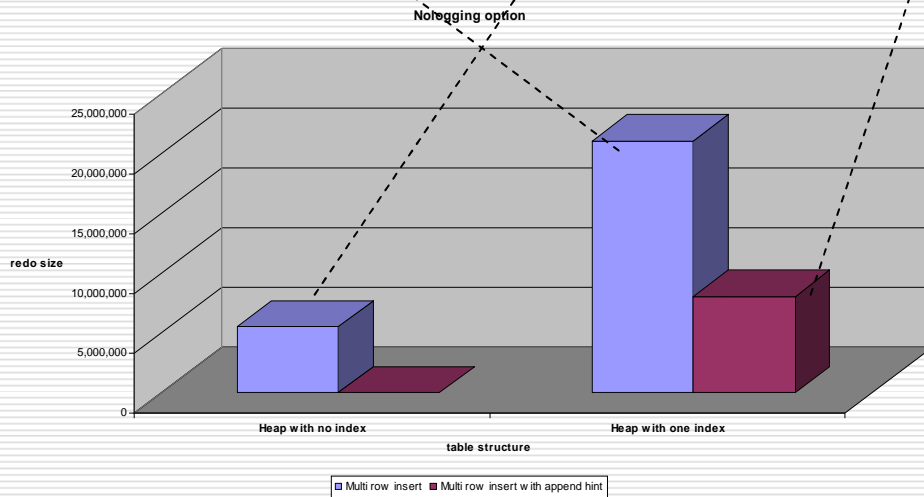
Nologging inserts-Multi row



Nologging..

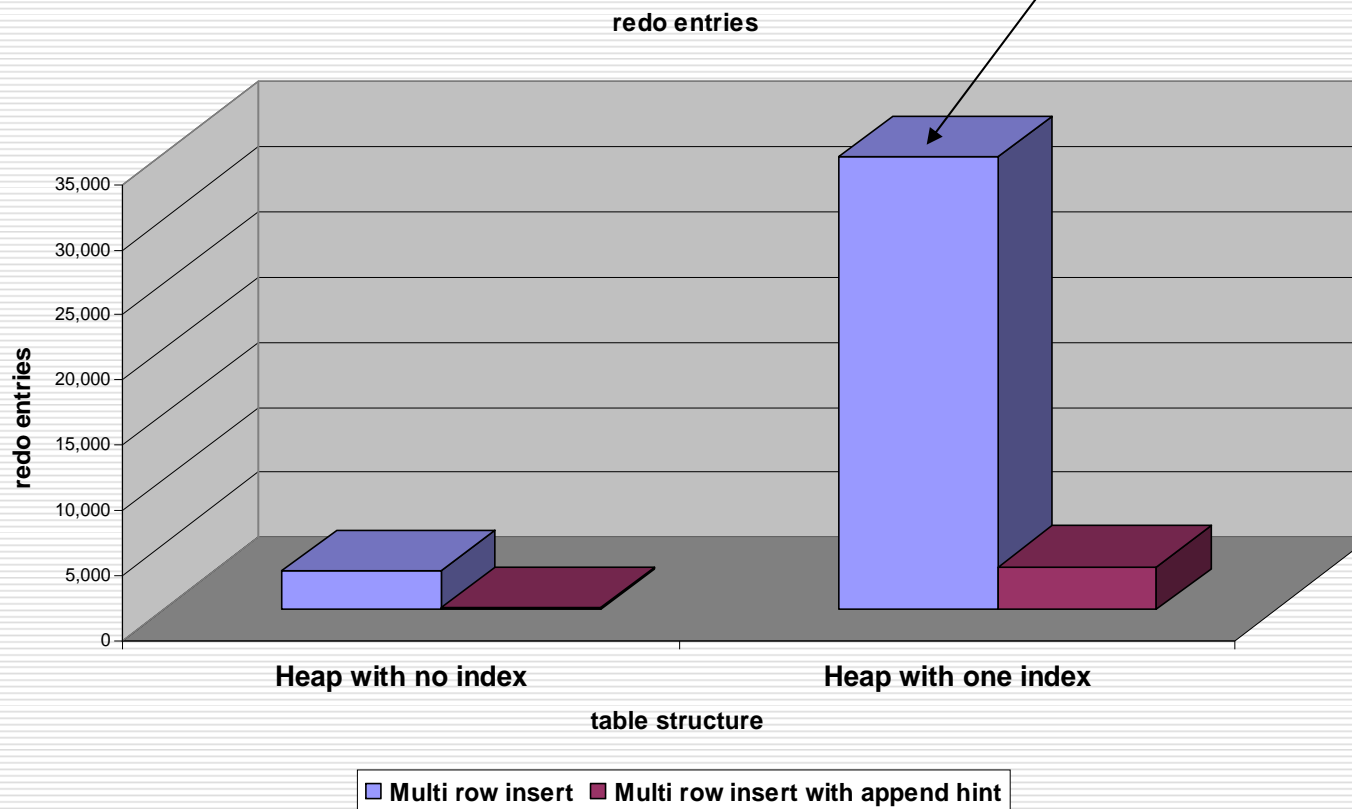
Redo size for table+index 21.1MB
WHY?

Redo size for table without index, conventional insert (5.5 MB)
+
Redo size for table+index WITH append (8 MB)



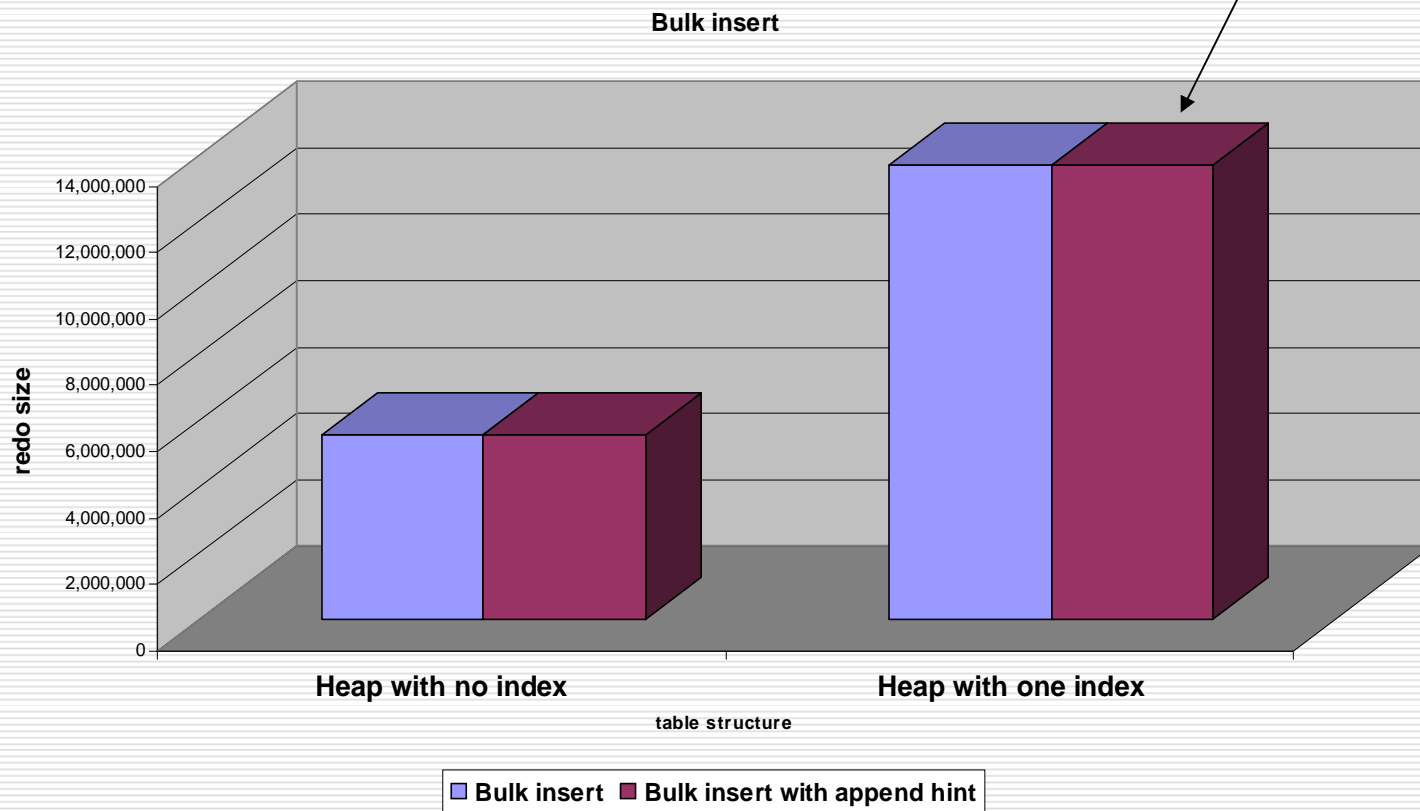
Nologging..

Too many redo records:
35,000 redo records for
80,000 rows!



Nologging – Bulk inserts

Nologging not honored for bulk inserts

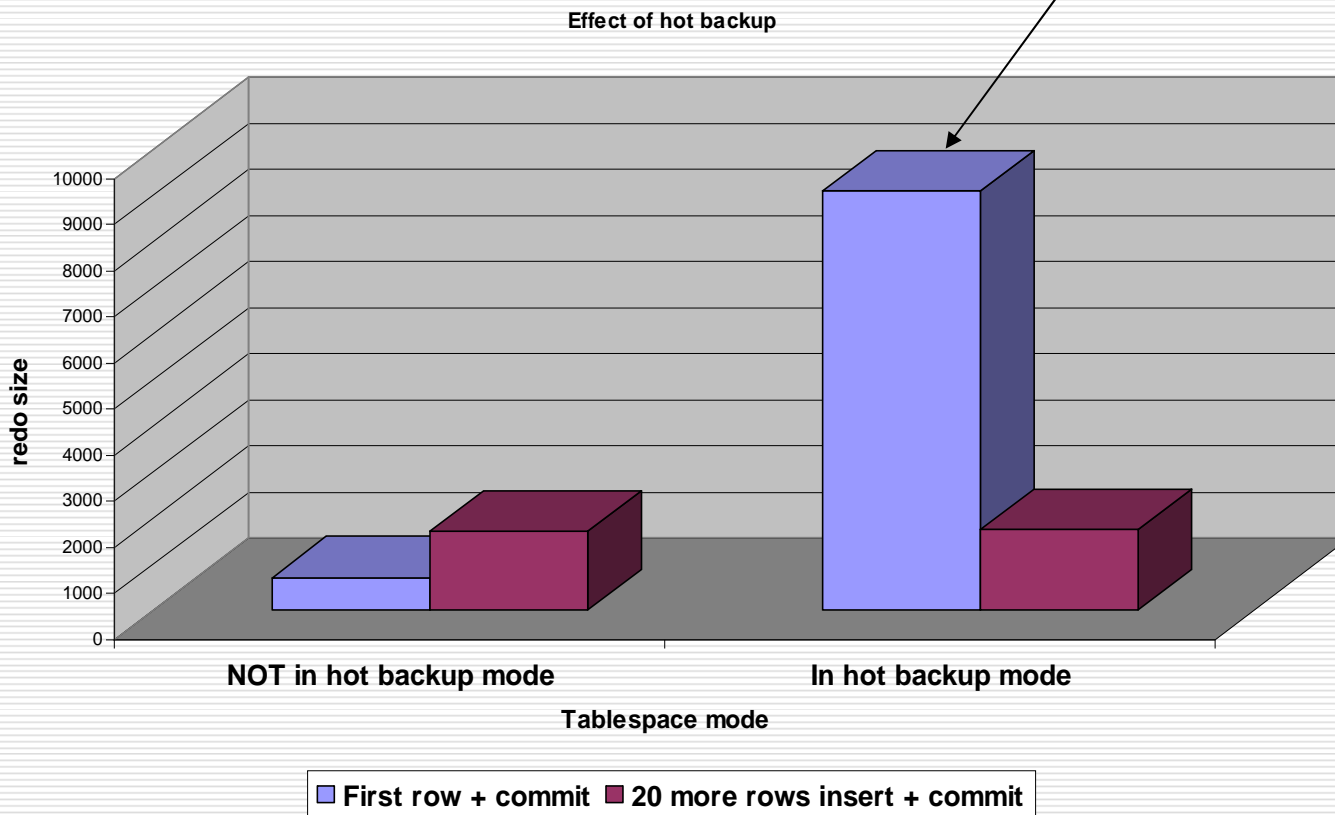


Effect of hot backup!

- During hot backup mode:
 - first change to a block generates full block image redo.
 - Avoids “split blocks” issue.
 - Reduce DML activity while the tablespaces are in hot backup.
 - RMAN does not suffer from this issue!
-

Effect of hot backup

First change to a block generates full block image, in hot backup mode



Effect of hot backup

8722 bytes

REDO RECORD - Thread: 1 RBA: 0x0000ba.00000002.0010 LEN: **0x2050** VLD: 0x05
SCN: 0x0000.00363d71 SUBSCN: 40 01/10/2006 16:05:30
CHANGE #1 TYP: 3 CLS: 1 AFN: 4 DBA: 0x0101868a SCN: 0x0000.00363d6a SEQ: 5
OP: 18.1

Log block image redo entry

Dump of memory from 0xFFFFFFFF72180420 to 0xFFFFFFFF72182408

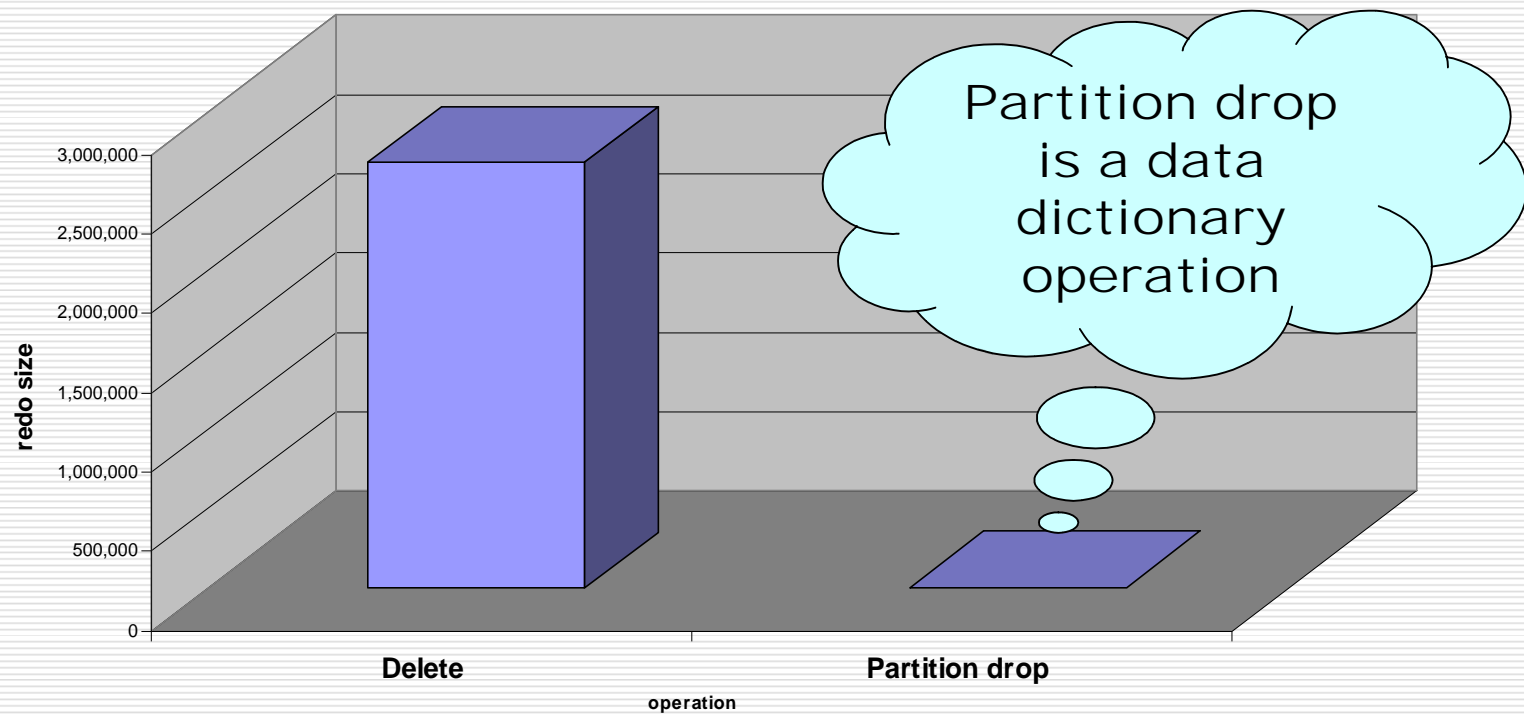
FFFFFFFF72180420	0103001B	00003574	00363D68	00000000	[.....5t.6=h....]
FFFFFFFF72180430	1F020300	00000000	00040005	0000019C	[.....]
FFFFFFFF72180440	00800EE7	03881E00	20050000	00363D6A	[..... 6=j]
FFFFFFFF72180450	00000000	00000000	00000000	00000000	[.....]
FFFFFFFF72180460	00000000	00000000	00010005	FFFF001C	[.....]
FFFFFFFF72180470	1E841E68	1E680000	00051E84	1EBD1EF6	[...h.h.....]

Partition drop vs delete

- Application purges old rows regularly
 - Using frequently a date criteria
 - Using some other key columns
 - Massive delete generates
 - Enormous amount of redo!!
 - Use partition drop techniques instead of deletes!
-

Partition drop vs delete

Delete vs partition drop



Unique vs non-unique

- Unique and non-unique indices have slightly different block structures..
 - Non-unique indices are implemented by appending rowid to the column list..
 - In a sense, even non-unique indices are unique internally, since rowid is unique!!
-

Unique vs non-unique

Block dump of unique index:

row#1[781] flag: ----S-, lock: 2, len=20, data:(6): 01 00 e3 49
00 3f

col 0; len 2; (2): c1 02

col 1; len 2; (2): c1 05

col 2; len 2; (2): c1 2c

col 3; len 2; (2): c1 04

Block dump of non-unique index:

row#0[7636] flag: -----, lock: 0, len=21

col 0; len 2; (2): c4 02

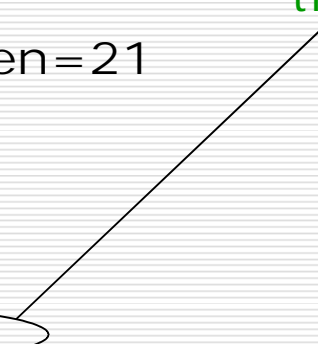
col 1; len 2; (2): c1 02

col 2; len 2; (2): c1 02

col 3; len 2; (2): c1 02

col 4; len 6; (6): 01 00 00 92 00

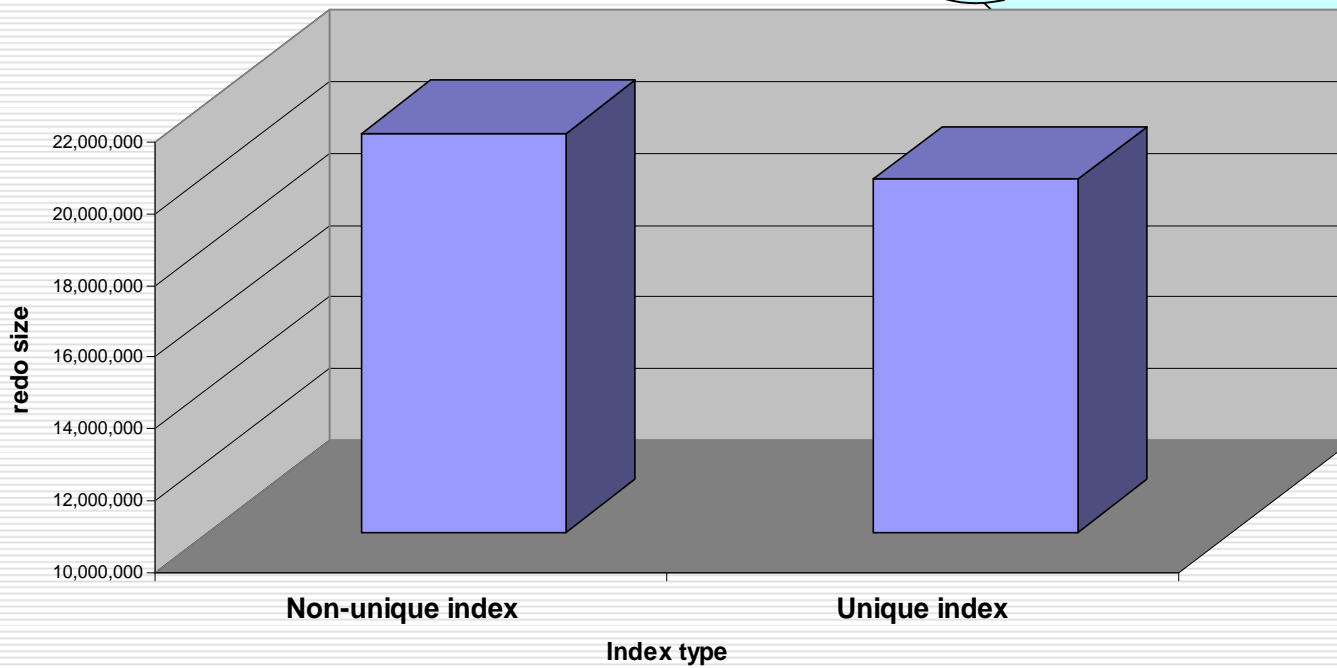
Row identifier
appended to
the column list



Unique vs non-unique

Unique : 19.8 MB
Non-unique: 21 MB

uniq vs nonunique



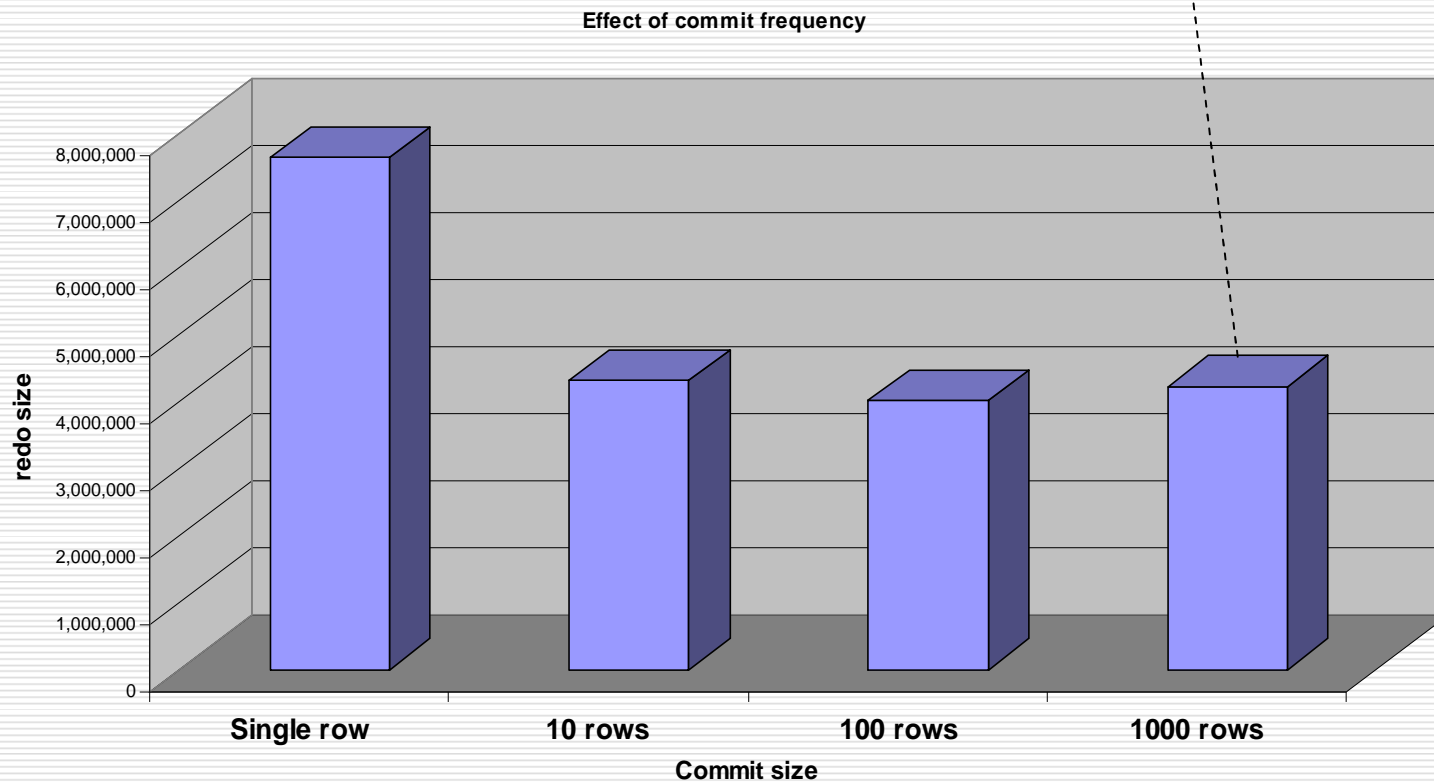
Commit frequency

- Commit can generate a separate redo record or can be included as a change vector, with other change vectors.
 - If included with other vectors, overhead is 72 bytes in 10g.
 - For a separate record, size is 140 bytes
 - Frequent commits increases redo size.
-

Commit frequency

For 1000 & 10000 rows, there is a minor increase in redo!

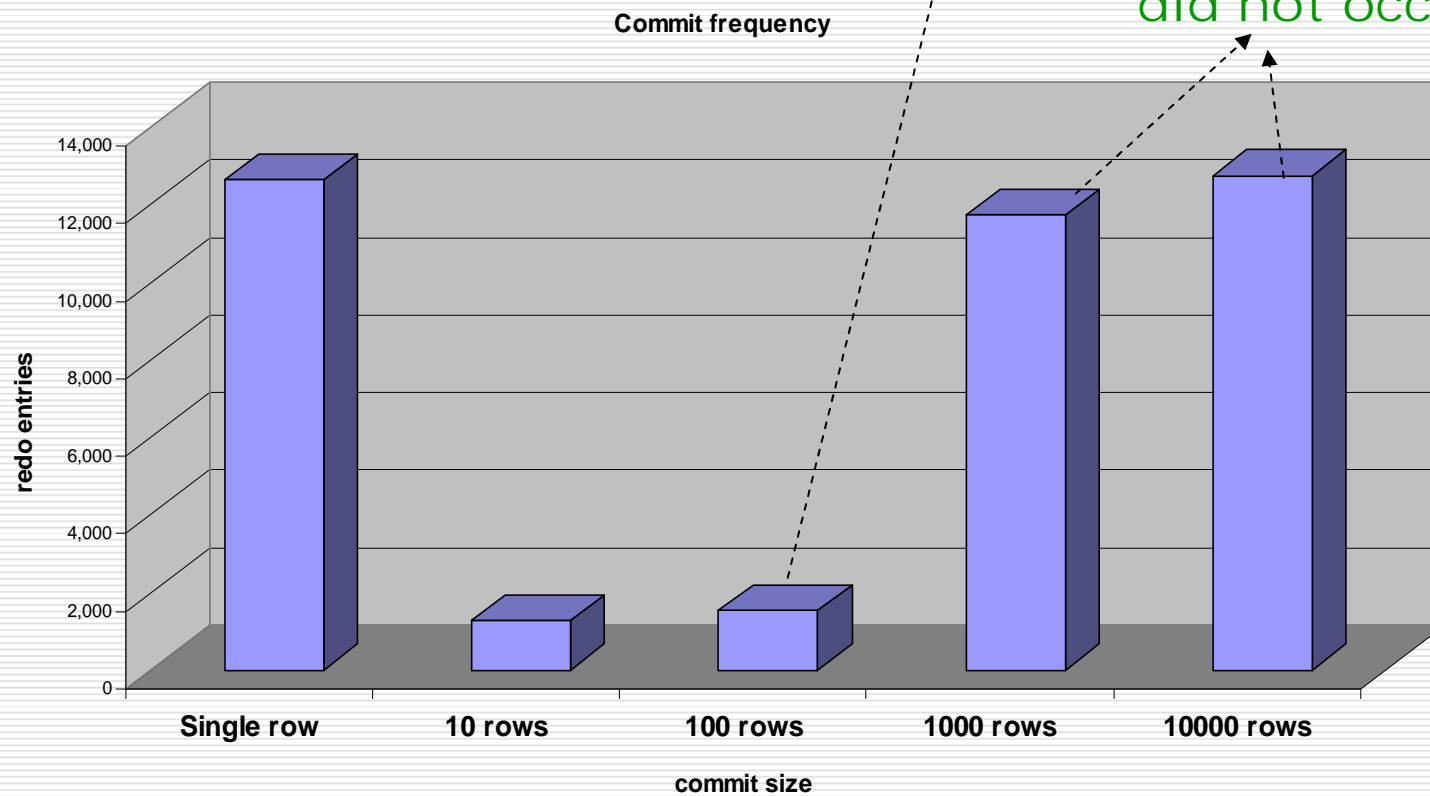
WHY?



Commit frequency

Every 10th redo record grouped 90 row changes

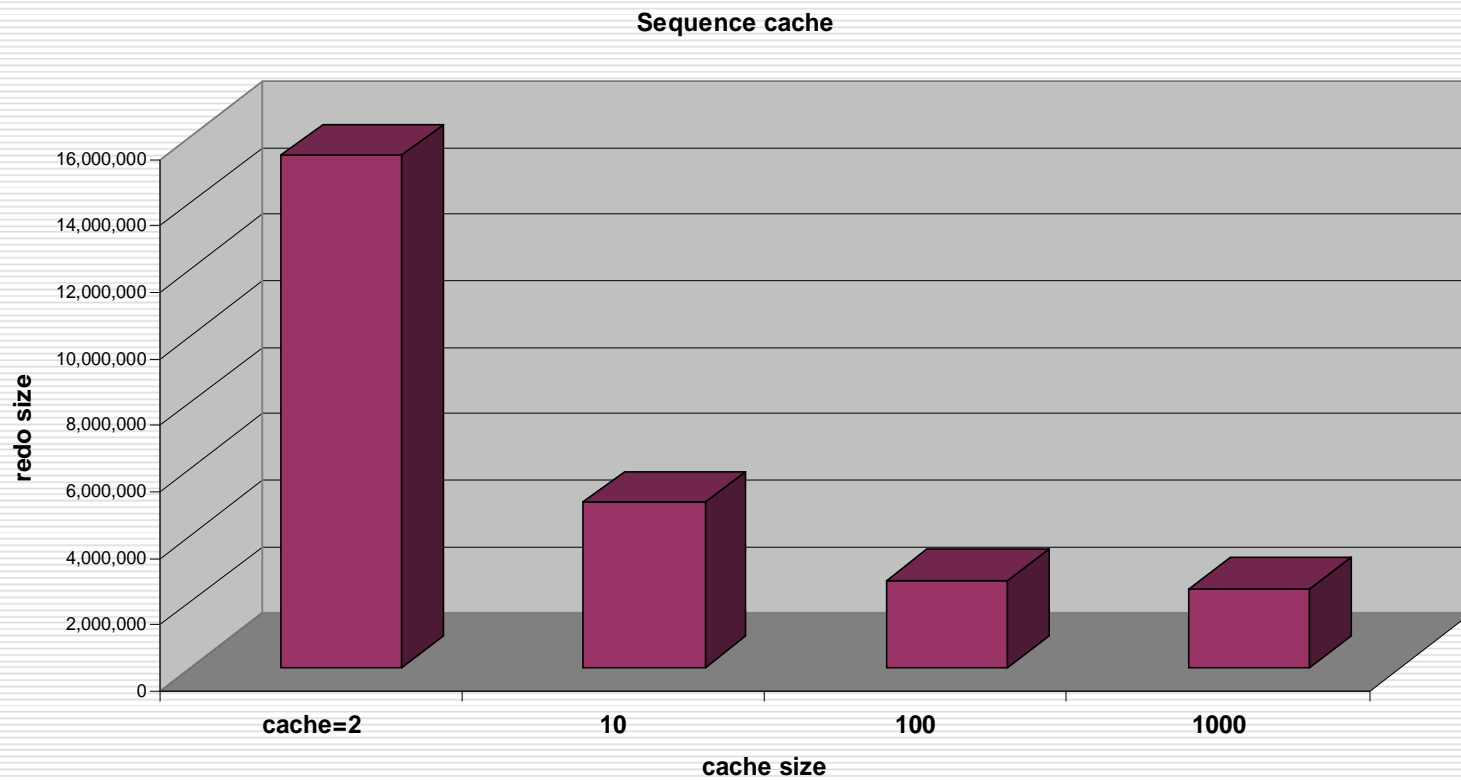
Above optimization did not occur!



Impact of sequence cache

- ❑ Oracle caches the sequence values in the instance memory.
 - ❑ Sys.seq\$ has permanent record of the sequence value.
 - ❑ If the # of values cached for a sequence is exhausted, seq\$ is updated and more values cached.
 - ❑ If the cache is set to small, this can lead to increased redo size.
-

Impact of sequence cache



Conclusion

- ❑ Discussed redo internals
 - ❑ Detecting excessive redo generation
 - ❑ Techniques to reduce redo
 - ❑ Test. Test.. Test...
-

Q & A

Questions ?
