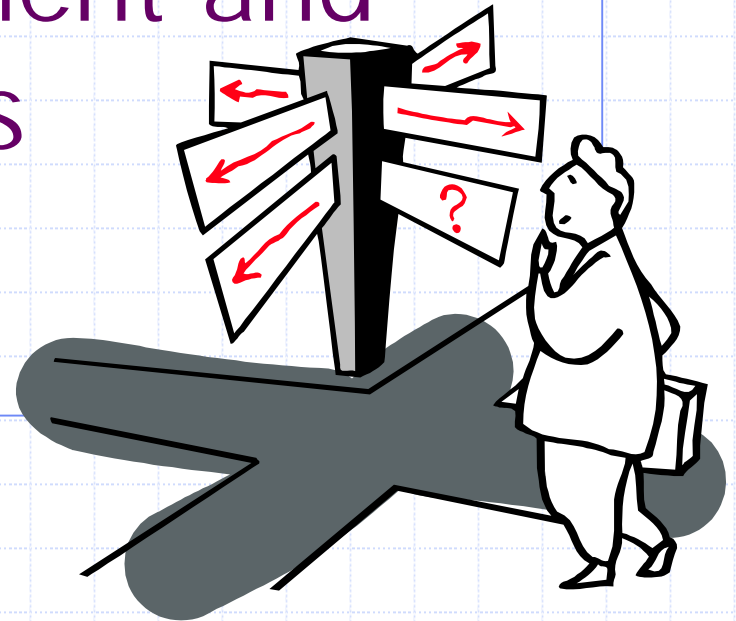


# Rollback Segment and Undo Internals

Paper 542

IOUG-A Live 2002

Daniel W. Fink



# Undo Internals

- ◆ Methodology
- ◆ Architecture Review
- ◆ UNDO Entries
- ◆ Administration
- ◆ What's 'New' with 9i

# Notes

- ◆ This presentation contains undocumented commands
  - These commands are not guaranteed to work on all versions and platforms
- ◆ Block dumps are not presented in original format
  - Important items are noted
  - Format changed for presentation purposes
- ◆ Paper contains additional information
- ◆ Issues well covered by other sources are not presented

# What is UNDO? (Survey)

- ◆ Copies of blocks changed by transactions, both uncommitted and committed
- ◆ Copies of database rows that are being updated, perhaps at the block level
- ◆ Snapshot of transactions that have occurred since the last commit
- ◆ Logical information stored in memory and not physically stored in a file
- ◆ Before image copy of data in a transaction
- ◆ System undo data allowing non-committed transactions to be rolled back

# What is UNDO? (Documents)

- ◆ "A rollback segment records the old values of data that was changed by each transaction (whether or not committed). Rollback segments are used to provide read consistency, to roll back transactions, and to recover the database." (Oracle8i Concepts)
- ◆ "They [rollback segments] capture the "before image" of data as it existed prior to the start of a transaction... A rollback segment entry is the set of before-image data blocks that contain rows that are modified by a transaction." (Oracle8 DBA Handbook)

# What is UNDO?

- ◆ Information necessary to reverse a transaction or to reconstruct a read consistent view.
- ◆ The contents of an undo entry include:
  - data column(s) address
  - transaction operation
  - old value(s) of the data.

# Methodology

- ◆ Identify components and processes of interest
- ◆ Create clean environment
- ◆ Perform single operation in isolation
- ◆ Identify changed components
- ◆ Conclusion
  - IF only single operation has occurred
  - THEN changes are the result of that operation

# Methodology

- ◆ Transaction will cause 4 changes (w/o index)
  - Data block
  - Rollback Segment
  - Redo Entry
    - ◆ 2...1 for data block, 1 for rbs
- ◆ Only data and rbs of interest
- ◆ Keep blocks contiguous for ease of monitoring

# Architecture Review

- ◆ Transaction

- ◆ Storage

- Segment

- Data Block

- UNDO

# Transaction

## ◆ Principles

- Active UNDO is never overwritten
- Never return dirty data
- Transaction UNDO cannot share a block with another active transaction

◆ Transaction information can be viewed in `v$transaction`

# Transaction

- ◆ Bind to a Rollback/UNDO Segment
- ◆ Allocate a slot in the Transaction Table
- ◆ Allocate a block for UNDO
- ◆ Update Interested Transaction List
- ◆ Save old value in UNDO block(s)
- ◆ Change data block to new value

# Storage

- ◆ Segment
- ◆ Data Block
- ◆ UNDO

# Segment

## ◆ Segment Header in Extent 0

- High water mark
- Extent Map

## ◆ Data begins in

- Extent 1 – table/index
- Extent 2 – undo
  - ◆ Why? – Not sure....
  - ◆ Not always used for wraps

# Data Block – Block Header

## ◆ Interested Transaction List (ITL)

- 1 per transaction modifying a row in the block
- Contents
  - ◆ Xid – Transaction ID
  - ◆ Uba – Undo Block Address
  - ◆ Lck – Lock Status

## ◆ Number of slots determined by

- INITRANS – initial created
- MAXTRANS – maximum number ever

# Data Block – Block Header

Block header dump: 0x01c00007

Object id on Block? Y

seg/obj: 0x565a csc: 0x00.7ae7c itc: 1 flg: 0 typ: 1 -  
DATA

fsl: 1 fnx: 0x0 ver: 0x01

Itl 0x01

Xid xid: 0x0003.000.00000012

Uba uba: 0x02000004.0006.09

Flag ----

Lck 9

Scn/Fsc fsc 0x0000.00000000

# Data Block – Block Header

## ◆ Xid

- 0x003.000.00000012
  - ◆ 0x003 – Undo Segment Number
  - ◆ 000 – Transaction Table Slot Number
  - ◆ 00000012 – Wrap

## ◆ UBA

- 0x02000004.0006.09
  - ◆ 0x02000004 – Address of the last undo block used
  - ◆ 0006 – Sequence
  - ◆ 09 – Last Entry in UNDO record map

# Data Block – Row Header

```
block_row_dump:
```

```
tab 0, row 0, @0x1e4c
```

```
t1: 44 fb: --H-FL-- lb: 0x1 cc: 8
```

```
col 0: [ 3] c2 4b 64
```

```
col 1: [ 5] 41 4c 4c 45 4e
```

```
col 2: [ 8] 53 41 4c 45 53 4d 41 4e
```

```
col 3: [ 3] c2 4d 63
```

```
col 4: [ 7] 77 b5 02 14 01 01 01
```

```
col 5: [ 3] c2 12 3d
```

```
col 6: [ 2] c2 04
```

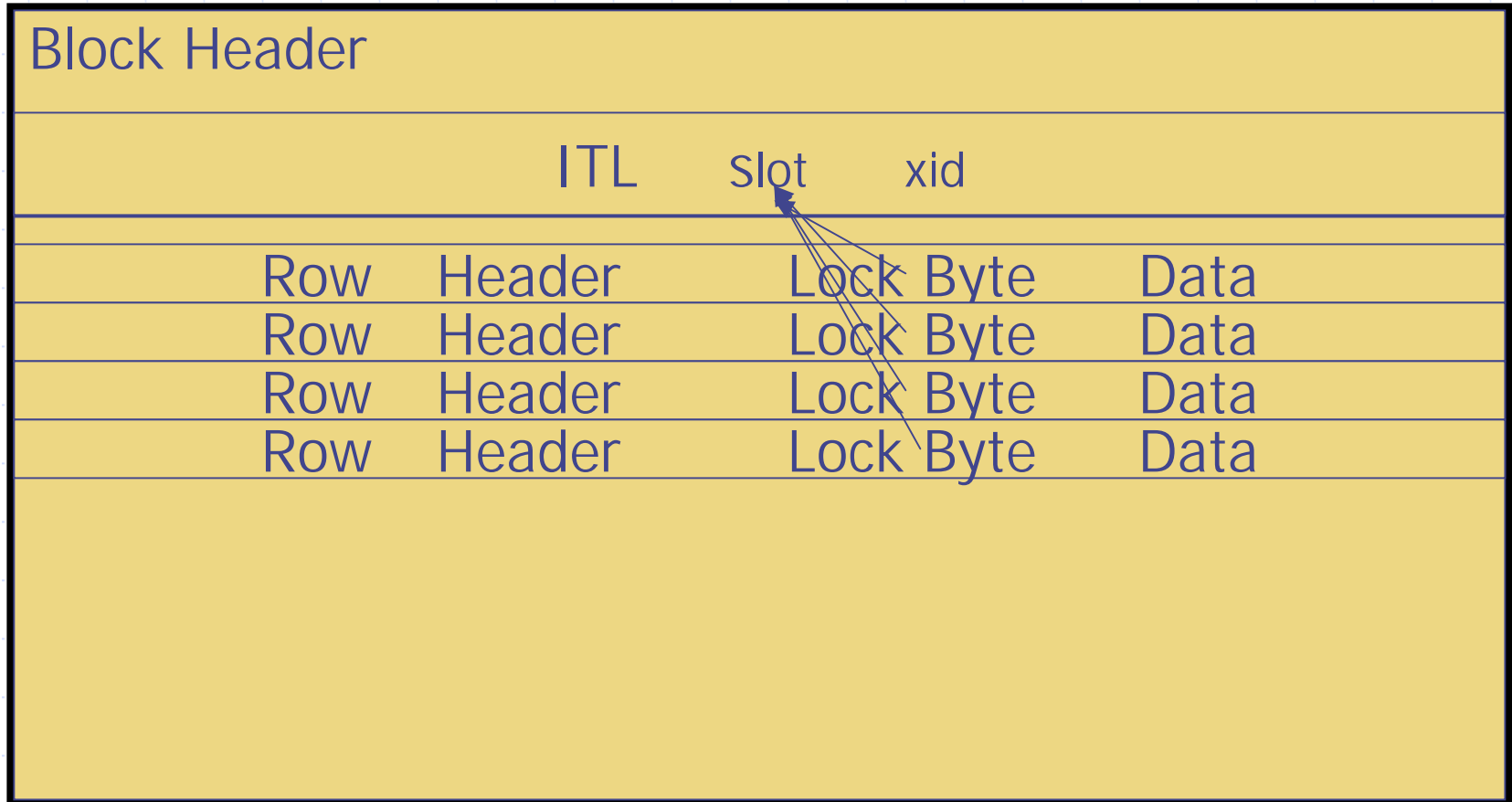
```
col 7: [ 2] c1 1f
```

# Data Block – Row Header

## ◆ Lock Byte

- Slot in ITL that currently locks row
- Only 1 transaction can modify a row at a time
- Multiple rows can be modified at one time by multiple transactions

# Data Block - Summary



# UNDO Architecture

- ◆ Transaction Table
- ◆ Undo Block Header
- ◆ Undo Entry

# UNDO Segment Header

```
Start dump data blocks tsn: 7 file#: 8 minblk 2 maxblk 2
buffer tsn: 7 rdba: 0x02000002 (8/2)
scn: 0x0000.0007ae92 seq: 0x01 flg: 0x00 tail: 0xae920e01
frmt: 0x02 chkval: 0x0000 type: 0x0e=KTU UNDO HEADER W/UNLIMITED
EXTENTS
TRN TBL::
```

index	state	cflags	wrap#	uel	scn	dba
0x00	10	0xc0	0x0012	0x0000	0x0000.0007ae92	0x02000004
0x01	9	0x00	0x0011	0x0002	0x0000.00075ff4	0x00000000
.....						
0x60	9	0x00	0x0011	0x0061	0x0000.00075ff4	0x00000000
0x61	9	0x00	0x0011	0xffff	0x0000.00075ff4	0x00000000

# Transaction Table

- ◆ In UNDO Segment Header (block 0)
- ◆ Contains pointer to last UNDO record for a transaction
- ◆ Number of slots is block size dependent

# UNDO Block Header

```
Start dump data blocks tsn: 7 file#: 8 minblk 4 maxblk 4
buffer tsn: 7 rdba: 0x02000004 (8/4)
scn: 0x0000.0007ae92 seq: 0x0a flg: 0x00 tail: 0xae92020a
frmt: 0x02 chkval: 0x0000 type: 0x02=KTU UNDO BLOCK
```

```
*****
```

```
UNDO BLK:
```

```
xid: 0x0003.000.00000012 seq: 0x6 cnt: 0x9 irb: 0x9 icl: 0x0 flg: 0x0000
```

Rec Offset	Rec Offset	Rec Offset	Rec Offset	Rec Offset
0x01 0x1f80	0x02 0x1f20	0x03 0x1ec8	0x04 0x1e70	0x05 0x1e18
0x06 0x1dc0	0x07 0x1d68	0x08 0x1d10	0x09 0x1cb8	

# UNDO Block Header

- ◆ XID – Transaction ID

- ◆ Undo Entry Map

- Rec – slot/index of each undo entry
- Offset – location of each entry

# UNDO Entry

- ◆ Information to UNDO a transaction
- ◆ Opposite logic
  - INSERT – DELETE
  - DELETE – INSERT
  - UPDATE – UPDATE w/old values
- ◆ Identify object and row (slot)
- ◆ Old value of each changed column
- ◆ UNDO Entry address of previous change in same transaction

# UNDO Entry – Single Column Update

```
* Rec #0x3  slt: 0x00  objn: 22101(0x00005655)  objd:
  22101  tblspc: 6(0x00000006)
op: 0x02  ver: 0x01
op: C  uba: 0x02000004.0005.02
KDO Op code: URP  xtype: XA  bdba: 0x01c00007  hdba:
  0x01c00006
itli: 1  ispac: 0  maxfr: 4863
tabn: 0  slot: 1(0x1)  flag: 0x2c  lock: 0  ckix: 0
ncol: 8  nnew: 1  size: 0
col 5: [ 3]  c2 1d 33
```

# UNDO Entry – Multiple Column Update

```
* Rec #0xb  slt: 0x01  objn: 22101(0x00005655)  objd:
  22101  tblspc: 6(0x00000006)
*op: 0x02  ver: 0x01
op: C  uba: 0x02000004.0005.0a
KDO Op code: URP  xtype: XA  bdba: 0x01c00007  hdba:
  0x01c00006
itli: 1  ispac: 0  maxfr: 4863
tabn: 0  slot: 1(0x1)  flag: 0x2c  lock: 0  ckix: 0
ncol: 8  nnew: 2  size: -1
col  5: [ 3]  c2 20 24
col  7: [ 2]  c1 1f
```

# UNDO Entry – Insert a row

```
* Rec #0x12  slt: 0x02  objn: 22101(0x00005655)  objd:
  22101  tblspc: 6(0x00000006)
uba: 0x02000004.0005.0a  ctl max scn: 0x0000.00075f77
  prv tx scn: 0x0000.00075f77
op: 0x04  ver: 0x01
op: L  itl: scn: 0x0003.001.00000010  uba:
  0x02000004.0005.11
                                flg: C---  lkc: 0  scn:
  0x0000.00075ff0
KDO Op code: DRP  xtype: XA  bdba: 0x01c00007  hdba:
  0x01c00006
itli: 1  ispac: 0  maxfr: 4863
tabn: 0  slot: 8(0x8)
```

# UNDO Entry – Delete a row

```
* Rec #0x13  slt: 0x03  objn: 22101(0x00005655)  objd: 22101
  tblspc: 6(0x00000006)
uba: 0x02000004.0005.12  ctl max scn: 0x0000.00075f77  prv tx
  scn: 0x0000.00075f77
op: 0x04  ver: 0x01
op: L  itl: scn: 0x0003.002.00000010  uba: 0x02000004.0005.12
      flg: C---  lkc: 0  scn:
      0x0000.00075ff2
KDO Op code: IRP  xtype: XA  bdba: 0x01c00007  hdba:
      0x01c00006
itli: 1  ispac: 0  maxfr: 4863
tabn: 0  slot: 8(0x8)  size/delt: 33
fb: --H-FL--  lb: 0x0  cc: 8
null: ---N--N-
col  0: [ 3]  c2 64 64
...
col  7: [ 2]  c1 0b
```

# UNDO Entry

- ◆ 1<sup>st</sup> Entry in Transaction contains
  - UNDO Information
  - ITL of previous transaction if overwriting ITL Slot
  - 1<sup>st</sup> UNDO Entry Address of previous transaction in block

# UNDO Segments and Entry - Summary

## Undo Segment Header

Transaction Table [index state scn dba]

## UNDO Block

Entry Map [rec offset]

Entry [uba values]

Entry [uba values]

Entry [uba values]

ITL Entry of previous Tx  
Previous UBA  
Previous UBA

# Read Consistent

- ◆ Undo Chain
- ◆ Block Cloning

# UNDO Chain

- ◆ All entries related to a single transaction form an UNDO Chain
- ◆ The chain represents a single coherent state of a block (or blocks)
  - Must include ITL entry if it is the 1<sup>st</sup> entry of a transaction
- ◆ May include a link to the previous transaction in the undo block
  - ITL represents last undo entry
  - UBA field in undo entry lists 1<sup>st</sup> entry of previous transaction

# Block Cloning

- ◆ UNDO entries are used to create a coherent version of the block
- ◆ A block is at a coherent state when all changes related to a single transaction have been applied or removed
- ◆ Each coherent version of the block will exist in the database block buffer cache

# Building a Read Consistent View

1. Read the Data block
2. Read the row header
3. Check the lock byte to determine if there is an ITL entry
4. Read the ITL to determine the transaction ID(txid).

# Building a Read Consistent View

5. Read the Transaction Table. If the transaction has been committed and has a System Commit Number less than the query's System Change Number, cleanout the block and move on to the next data block and step 1
6. Read the last undo block indicated

# Building a Read Consistent View

7. Compare the block transaction id with the transaction table txid. If the txid in the undo block does not equal the txid from the transaction table, then signal an ORA-1555 (Snapshot Too Old).
8. If the txids are identical, clone the data block in memory. Starting with the head undo entry, apply the changes to the cloned block.

# Building a Read Consistent View

9. If the tail undo entry (the last one read) indicates another data block address, read the indicated undo block into memory. Repeat steps 7 & 8 until the first record does not contain a value for the data block address.
10. When there is no previous data block address, the transaction has been undone.

# Building a Read Consistent View

## 11. If the undo entry contains

- a) A pointer to a previous transaction undo block address, read the txid in the previous transaction undo block header and read the appropriate transaction table entry. Return to Step 5.
- b) an ITL record, restore the ITL record to the data block. Return to Step 4.

# Building a Read Consistent View - Summary

db block buffer cache

Original	tx1	tx2
tx3	tx4	tx5
Curr		

undo chain

tx1
tx2
tx3
tx4
tx5

# Administration

- ◆ Monitoring
- ◆ Wraps, Shrinks & Extends
- ◆ Errors
- ◆ Delayed block cleanout

# Monitoring

◆ v\$rollstat

◆ dba\_rollback\_segs

# v\$rollstat

- ◆ View of performance of rollback segment since instance startup
- ◆ Space Management
  - $\text{optsize} < \text{hwmsize}$  – RBS optimal setting too small
  - $(\text{shrinks or extends}) > \text{wraps}$  – RBS too small
- ◆ Performance columns
  - gets – High number = high activity
  - waits – High number = too few segments

# Other views

## ◆ dba\_rollback\_segs

- Defined structure of rollback segment

## ◆ v\$rollname

- relationship between rollback segment names (defined in init.ora) and undo segment number

# Wraps, Shrinks & Extends

## ◆ Wraps

- Move into an existing extent
- Good statistic...no space management issues

## ◆ Shrinks

- Deallocation of extents
- 2<sup>nd</sup> transaction to discover over extended segment causes shrink

## ◆ Extends

- Allocation of new extent
- Bad statistic...space management

# Errors

- ◆ ORA-01555 – Snapshot Too Old
- ◆ Space Management

# ORA-1555

- ◆ Data required to build read consistent view is missing
- ◆ Undo Entry has been
  - Overwritten by another transaction
  - Existed in a deallocated block from a shrink operation
- ◆ Indicates
  - Segments too small to hold entries for sufficient time
  - Long running query on a high activity OLTP system

# ORA-1555

## ◆ Fetch Across Commit

- Long Running PL/SQL loop
  - ◆ Data in cursor must maintain consistency
  - ◆ As of time of 1<sup>st</sup> read
- The cursor selects data, makes a change and commits

## ◆ Solutions

- Commit less frequently (requires more undo space)
- Add an 'ORDER BY' clause (data is read into TEMP segment)

# ORA-1555

## ◆ Overwritten Transaction Table

- Finite number of slots
- If slot is overwritten by new transaction, link to undo entries are lost

## ◆ Solutions

- Increase block size (requires database rebuild)
- Increase number of rollback segments

# ORA-1555

## ◆ Shrinking

- Segment has extended beyond the OPTIMAL setting
- 2<sup>nd</sup> transaction to discover this state will cause deallocation of extents above the OPTIMAL
  - ◆ As long as no active transactions are in any of these blocks
  - ◆ Also done by manual command or SMON

## ◆ Solutions

- Change OPTIMAL setting to more accurately reflect segment sizes

# Space Management

## ◆ ORA-1562 Unable to Extend

- Segment attempts to allocate new extent, but finds inadequate free space in tablespace data files

## ◆ Too Many/Too Large Transactions

- Insufficient space for all undo entries for concurrent transactions

## ◆ Too Few Transactions

- Round Robin Allocation
- Will not shrink until new transaction binds to it

# Delayed Block Cleanout

- ◆ When a transaction is committed, the change may have already been written to disk. Rereading and writing the block would be extra I/O.
- ◆ When a query finds a committed change, it updates the lock byte and ITL and writes the block to disk
- ◆ The change may never be written if the block is never accessed again

# Oracle9i Automatic Undo

- ◆ Architecture
- ◆ Maintenance and Operation
- ◆ New Views

# 9i Architecture

## ◆ UNDO tablespace

- Only 1 online per instance at one time
- Locally Managed
  - ◆ Bitmap extent management
- Autoallocate
  - ◆ 64k / 1m extents

# 9i Architecture

## ◆ Automatically created and managed UNDO Segments

- Up to 10 are created upon creation of tablespace
- Function of Sessions parameter
  - ◆ Roughly 1 UNDO Segment for 5 Sessions

# 9i Architecture

- 2 extents are created for each undo segment
  - ◆ Block 0 of Extent 0 is UNDO Header
- `_SYSSMUn$` naming convention

# 9i Creation

- ◆ Creating a database with System Managed Undo
  - Specify `undo_tablespace` in create database command
- ◆ Altering the database to use System Managed Undo
- ◆ Creating a new undo tablespace

# 9i Parameters

## ◆ UNDO\_MANAGEMENT

- auto – use system managed undo
- manual – use the traditional method of rollback segments

## ◆ UNDO\_TABLESPACE

- identifies the current undo tablespace
- Can be dynamically altered
- Only 1 undo tablespace can be allocated to an instance at any given time

# 9i Parameters

## ◆ UNDO\_SUPPRESS\_ERRORS

- Ignore 'set transaction use rollback segment rbs01' errors

## ◆ UNDO\_RETENTION

- The amount of time, in seconds, that undo information will be retained
- Not an absolute guarantee

# 9i Operation

- ◆ Each transaction is assigned to its own undo segment
  - If all undo segments in use by active transaction, create a new undo segment
  - If space does not allow for a new segment, assign to a segment currently in use

# 9i Operation

- ◆ When a transaction needs more space
  - Reclaim expired extents from current transaction
  - Acquire a free extent from the current undo segment, assuming free extents are available
  - Acquire a free extent from another undo segment, assuming no free extents are available
  - Extend the datafile
  - Allocate space from unexpired transactions in current undo segment
  - Allocate space from unexpired transactions in other undo segment
  - Raise ORA-01562 error

# 9i Views

- ◆ DBA\_UNDO\_EXTENTS
- ◆ V\$UNDOSTAT
- ◆ UNDO\$

# DBA\_UNDO\_EXTENTS

- ◆ Similar to `dba_rollback_segs`
  - Only shows system managed undo segments
  - Will not show SYSTEM rollback segment
- ◆ Segment Header is not indicated

# V\$UNDOSTAT

- ◆ 10 minute interval data on undo performance
  - begin\_time/end\_time – time interval boundaries
  - undoblks – number of undo block used
  - txncount – number of transactions
  - maxquerylen – longest query time in seconds
  - maxconcurrency – highest number of concurrent transactions
  - unxpstealcnt – number of attempts to obtain space by allocating unexpired extents from other transactions

# UNDO\$

◆ us# - UNDO Segment Number

◆ name – UNDO Segment Name

◆ status\$

- 1 invalid
- 2 defined but not is use
- 3 online
- 4 offline
- 5 needs recovery
- 6 partially available

# Conclusion

- ◆ UNDO is a critical transaction component
- ◆ Examination leads to in-depth knowledge of how Oracle really works
- ◆ For more information  
[www.optimaldba.com](http://www.optimaldba.com)  
[optimaldba@yahoo.com](mailto:optimaldba@yahoo.com)