Cats, Dogs, and ORA-01555

Tim Gorman
(tim@sagelogix.com)
Principal
SageLogix, Inc.

www.sagelogix.com
Agenda

• The eternal struggle
• How rollback segments work
• What’s the real cause of ORA-01555?
• What can be done about it?
• Guidelines for creating and tuning rollback segments
• Looking ahead: Oracle9i
Q: What do you call five-hundred DBAs at the bottom of the ocean?

Q: A priest, a DBA, and a developer are marooned in a life raft…
The eternal struggle

• Why the animosity?
  ➢ There are many reasons, but the standard error message for the ORA-01555 tends to make it flare up...

ORA-01555 snapshot too old: rollback segment number "###" with name "string" too small

**Cause:** Rollback records needed by a reader for consistent read are overwritten by other writers.

**Action:** Use larger rollback segments.

➢ Yowza! Don’t believe everything you read!
How rollback segments work

• What do they do?
  ➢ Store the “before-images” of data changes for a transaction, so that the “after-images” can reside in the DATA block
  ➢ Quiz question: Wouldn’t they be faster if they resided entirely in the SGA?

• Why?
  ➢ Enable transaction-level recovery (a.k.a. ROLLBACK)
  ➢ Permit read-consistency while the transaction is active (uncommitted) as well as after it has been committed
How rollback segments work

- Space is allocated in *extents*
- Transactions store *undo records* into *undo blocks*
- Each transaction stores control information in the *transaction table*, which resides in the *undo header* of the RBS
  - 8Kb blocks have 97 entries
- *Undo blocks* are provided on a *first-come, first-served* basis, moving sequentially around the *circular buffer* of the RBS
How rollback segments work?

EXTEND?
Allocate a new Extent?

Extent 35, inserted between 10 and 11

WRAP?
Into an existing Extent?

Extent 10

Extent 11
What is happening?

- **Steps of a transaction in Oracle:**
  - Transaction initiated via INSERT, UPDATE, or DELETE statement
  - Allocate rollback segment to transaction
    - First *round-robin* then *LRU* algorithm amongst online RBS
    - Oncetxn assigned to an RBS, there is no reconsideration
  - Allocate *slot* in *transaction table* in RBS header
    - *XID* consists of XIDUSN, XIDSLOT, XIDSEQ
    - V$TRANSACTION is a view into these data structures
      - V$ROLLNAME.USN and V$ROLLSTAT.USN joined via XIDUSN column
  - Acquire TX enqueue on XID and TM enqueue on object
What is happening?

• Steps of a transaction in Oracle (cont’d)
  5. Allocate *undo block(s)* to hold before-image info
     - Each undo block holds data for one and only one txn
     - Row changes stored in *undo records* in *undo blocks*
  6. Allocate *interested-transaction list (ITL)* entry in affected database block
     - Fields in ITL include:
       - ITL number
       - XID
       - Undo block address (UBA) of undo record in the RBS
       - Flag and lock bits for transaction status
       - Combination field to hold SCN (if txn is committed) or *free space credit* (FSC) if txn is not committed and txn causes row size to shrink (i.e. UPDATE or DELETE)
  7. Copy *before-image* data to undo records in undo blocks
Steps of txn initiation

Data segment:
- Allocate RBS
- Allocate txn tbl slot
- Allocate undo blk(s)
- Allocate ITL
- Copy before-image
- Insert after-image

Rollback segment:
- Block Header
- Txn Table
- Block Header
- Undo block
- Block Header
- Undo records

Data block
- ITLs
- Row Data
How rollback segments work

• Have you ever parked your car at Disney World?
  - Cars enter the lot single-file
  - Guided into spaces one-by-one, sequentially
  - When a row fills, the line is guided to another row

• Now, just imagine if:
  - A lone car is left in a row overnight and (as a result) the row must be skipped and a new row built *on the spot*?

• *Yes, very silly -- but please bear with me…*
How rollback segments work

- So, refining the example of the “amusement park parking lot”:
  - Everything runs smoothly if cars stay only a couple hours or less
  - Parking lot doesn’t “grow” due to “unusable” rows…
    - “Rows” get reused readily
  - But trouble, trouble, trouble if they stay overnight!
  - Parking lot staff are forced to “build” new rows…
    - Worst possible situation (from a space perspective) is one car left in each row…
How rollback segments work

• But wait! Let’s take it just one step further (groan!)…
  ➢ So far, we’ve only discussed the impact on space utilization…
  ➢ OK, OK, OK: now, just imagine that some teenagers had to hop from car-roof to car-roof across the parking lot (for some reason)…
    ➢ …wouldn’t missing cars present a problem?

(Yes, I’ve been watching my son play Nintendo…
  …you wouldn’t believe what story lines they come up with!)

www.sagelogix.com
How rollback segments work

- Try to think of it in terms of the imaginary parking lot:
  - If transactions are committed quickly, then there are no space problems
  - Rollback segments would never have to EXTEND
    - They’d just WRAP all the time!
      …round and round and round…(wheeeeee!)
  - But! Leave just one little transaction out there for a while…
    - What happens??????
• Now what about those car-surfing teen-agers?
  ➢ Yeah, I know – arrest ‘em!
  ➢ Or, make ‘em try to jump the gap (splat!)

• If transactions commit quickly, the leading-edge of transactions keeps wrapping into extents which are full of inactive (i.e. already committed) transactions
  ➢ What happens to queries (i.e. car-surfing delinquents) that need to use those now over-written undo blocks?
    ➢ Bingo! ORA-01555
      – Splat! “Bummer, dude!” “That’s so *rude*!”
What’s the real cause of ORA-01555?

- It is caused by the \textit{leading edge} of new transactions sweeping around the \textit{circular buffer} too quickly and over-writing blocks which are still needed.

What can be done about it?

- Option \#1: Make the RBS huge
  - Lots of extents seem to work well
- Option \#2: Don’t mix long-running queries with OLTP workload (i.e. workload scheduling)
- Option \#3: Try not to perform periodic commits
- Option \#4: Try including an ORDER BY
What can be done?

• Take the example of the batch process that is *stepping on itself*…
  - Main cursor loop of program
    - For each iteration, update/delete rows in one of the tables in the main cursor
    - Commit
    - Repeat

• Think of what is happening in the RBS
  - What would help in this example?
What can be done?

• Take the example of a long-running query that is being *stepped on* by another session
  ➢ Suppose an UPDATE statement updates one row -- *and commits immediately* -- which will be accessed 12 hours from now by a long-running query?

• Visualize what has occurred in the RBS
  ➢ What would help in this case?
What can be done?

• Package PREVENT1555 and shell script “prevent1555.sh”
  ➢ Stored procedure creates a small “dummy” transaction to create a *roadblock* in the specified rollback segment
  ➢ Must also specify the duration of the “dummy” transaction in minutes
  ➢ Shell script calls stored procedure for every online non-SYSTEM rollback segment

• Why might this be helpful?

  Please consider this a *last resort* for emergencies only…

• Online at http://www.EvDBT.com/tools.htm
Looking ahead

- Oracle9i introduces exciting new features
  - UNDO tablespaces
  - Pre-configured RBS within a locally-managed tablespace
    - Rollback segments managed by RDBMS
      - Not configurable
  - RBS extent reuse also constrained by UNDO_RETENTION parameter
    - Specified in seconds (default: 900, max: $2^{32}$)
    - Can be set in “init.ora” at instance STARTUP
    - Can be modified with ALTER SYSTEM SET
  - Do not use in 9.0.1.0, 9.0.1.1, or 9.0.1.2!!!
    - Serious bugs fixed in 9.0.1.3 and 9.2.0.1 and above
Give peace a chance

• DBAs
  ➢ Don’t hang up on developers who quote the ORA-01555 error message chapter and verse…
    ➢ After all, adding more RBS space *might* be the answer!

• Developers
  ➢ Don’t believe everything you read in the Oracle documentation!
  ➢ 500 Gb for each rollback segment *probably is* unreasonable

• Help is on the way in Oracle9i
• There are tricks to try in prior versions
Thank you!