



ORACLE®

Mughees A. Minhas
Principal Product Manager
Database Manageability
Oracle Corporation

**The Do's and Dont's of Space &
Undo Management:
Best Practices for Oracle Database 10g**

Agenda

- Space Management
 - Permanent Tablespace Management
 - Temporary Tablespace Management
- Undo Management
 - Manual vs. AUM
 - AUM Administration
- Proactive Problem Prevention and Automatic Error Handling

Permanent Tablespace Management Best Practices

Space Management Goals

- Space Utilization
 - Optimize space usage by eliminating/ minimizing fragmentation
- Performance
 - Optimize data access and transaction performance
- Choices that affect space utilization and performance are:
 - Extent management
 - Segment space management
 - Number of extents in tablespace
 - Size of extents in tablespace

Extent Management

- Options:
 - Dictionary Managed Tablespaces
 - Extents managed in dictionary tables
 - Fine control over extent sizes through storage parameters INITIAL, NEXT and PCTINCREASE
 - Locally Managed Tablespaces
 - Space managed locally by bitmaps in data file headers
 - Two extent management types
 - Auto-allocate : Extent size determined by database
 - Uniform: All extents of same size
- Poor extent management can lead to
 - External fragmentation
 - Poor performance

Extent Management

- External Fragmentation
 - Definition
 - Fragments of free extents not large enough for new allocations
 - Cause: Dictionary Managed Tablespaces
 - Improper choice of storage parameters (INITIAL, NEXT, PCTINCREASE)
 - Frequent creation/dropping of objects leaving numerous different size free extents
 - Impact
 - Poor space utilization
 - Space management performance can deteriorate by 10% to 20%

Extent Management

- Poor Performance
 - Cause: use of Dictionary Managed Tablespaces
 - Database wide serialization through ST enqueue
 - Metadata maintenance CPU and I/O intensive
 - Recursive SQLs to update metadata
 - Fragmentation itself also results in poor response time

Best Practice

- Use Locally Managed Tablespaces
 - Serialization of space management at file level
 - Space management faster by 100% to 200%
 - Eliminates external fragmentation
- Extent management types
 - Auto-allocate (**Recommended**): Extent size determined by database
 - Step-down algorithm minimizes fragmentation by lowering space allocation in multiples of 1M when desired size not available
 - Uniform: All extents of same size
 - Use when all segment have roughly same size
 - Segment growth rates are known precisely
 - Parallel direct loads not common

Best Practice

- Migrate to locally managed
 - Use PL/SQL package
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL
 - Migration done in-place, online
 - Multiple tablespaces can be migrated in parallel
 - Out-of-place migration if tablespace already fragmented

Segment Space Management

- Manual Segment Space Management (MSSM)
 - Free space managed using linked list of free blocks
 - Movement of blocks in freelist controlled by PCTFREE, PCTUSED
 - Buffer busy waits reduced by
 - FREELISTS: data blocks
 - FREELIST GROUPS: segment header block
- Automatic Segment Space Management (ASSM)
 - Segment space managed using bitmaps
 - Bitmaps stored in metadata blocks called bitmap blocks
 - Number of bitmap blocks dynamically grown as needed
 - PCTUSED, FREELISTS, FREELIST GROUPS don't have to be set

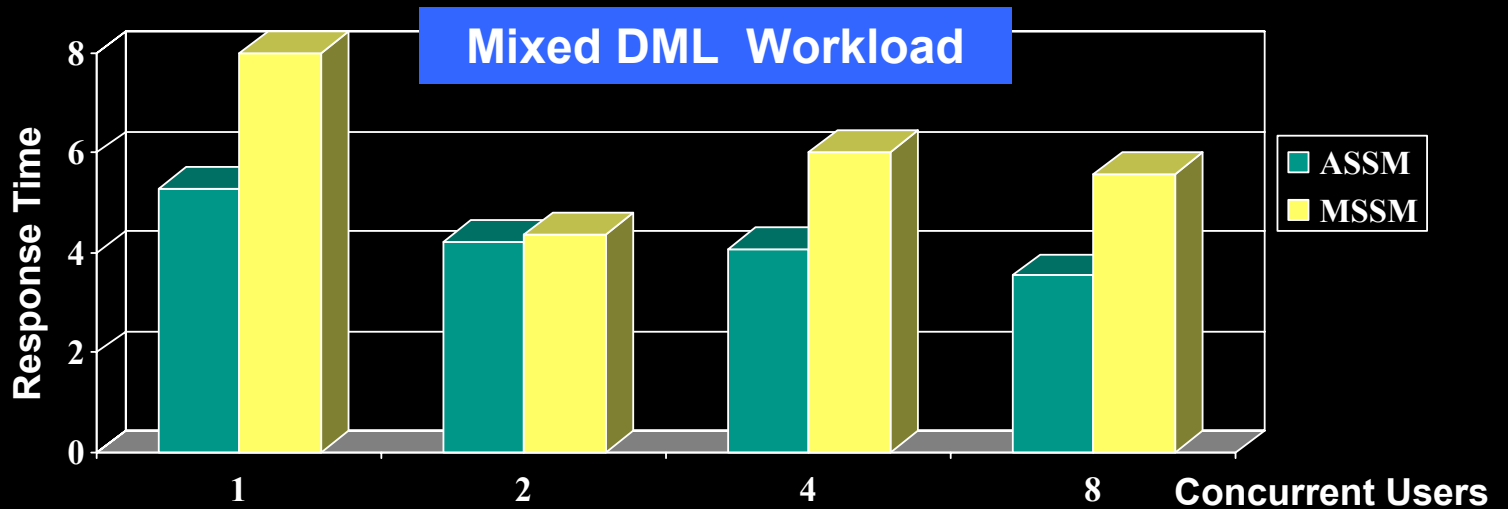
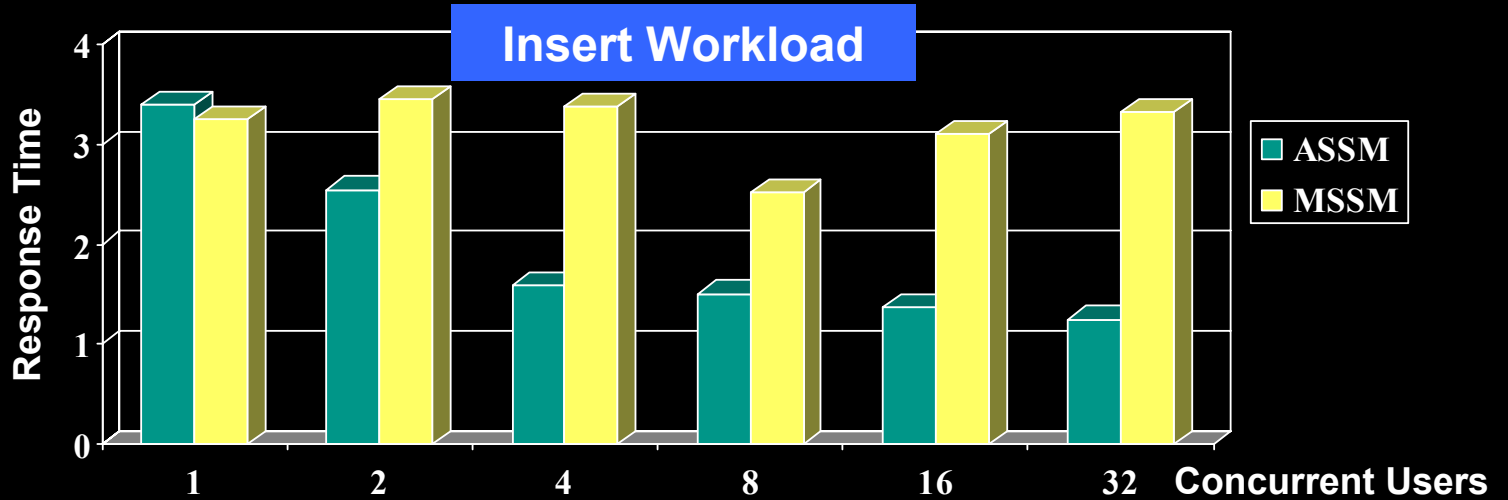
Segment Space Management

- Problems with MSSM
 - Can lead to internal fragmentation
 - FREELISTS parameter not dynamic
 - Contention affects performance
 - FREELIST GROUPS
 - Statically partition free space: In RAC space freed by one instance group cannot be used in other instances
 - Cannot be reconfigured without rebuilding the object
 - Does not adjust to varying loads

Best Practice

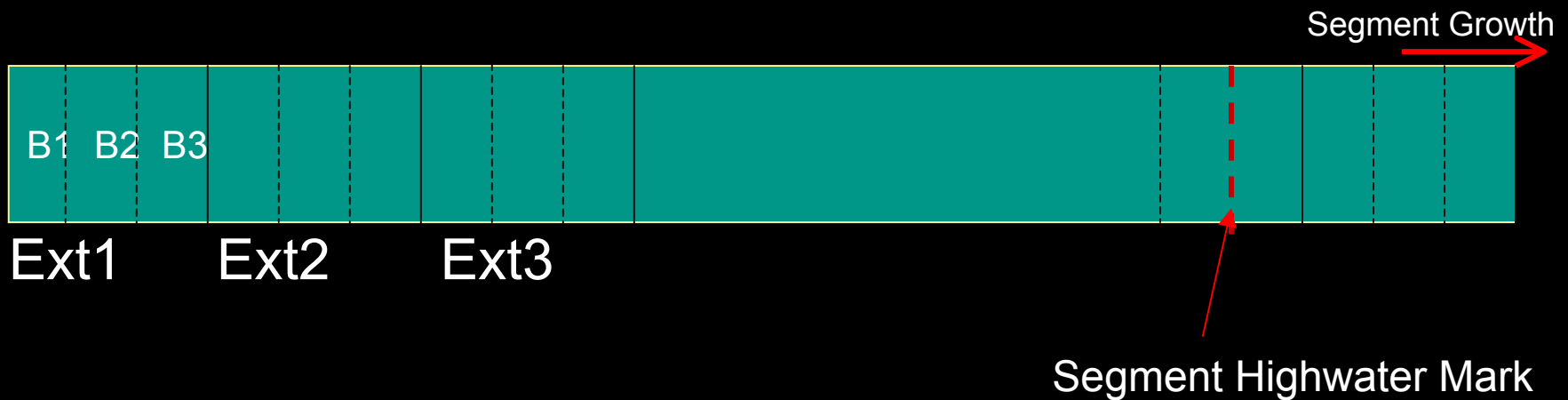
- Use Automatic Segment Space Management (ASSM)
 - Minimizes internal fragmentation
 - Contention on metadata blocks automatically managed
 - Easier configuration
 - Performance on par or better than **tuned** MSSM
 - Inter-instance data block contention reduced by dynamic instance affinity

ASSM vs. MSSM



Internal Fragmentation

- Definition: Fragmentation of space within a segment
 - Under-utilization of data under HWM



- Although minimized, can occur in ASSM tablespace as well

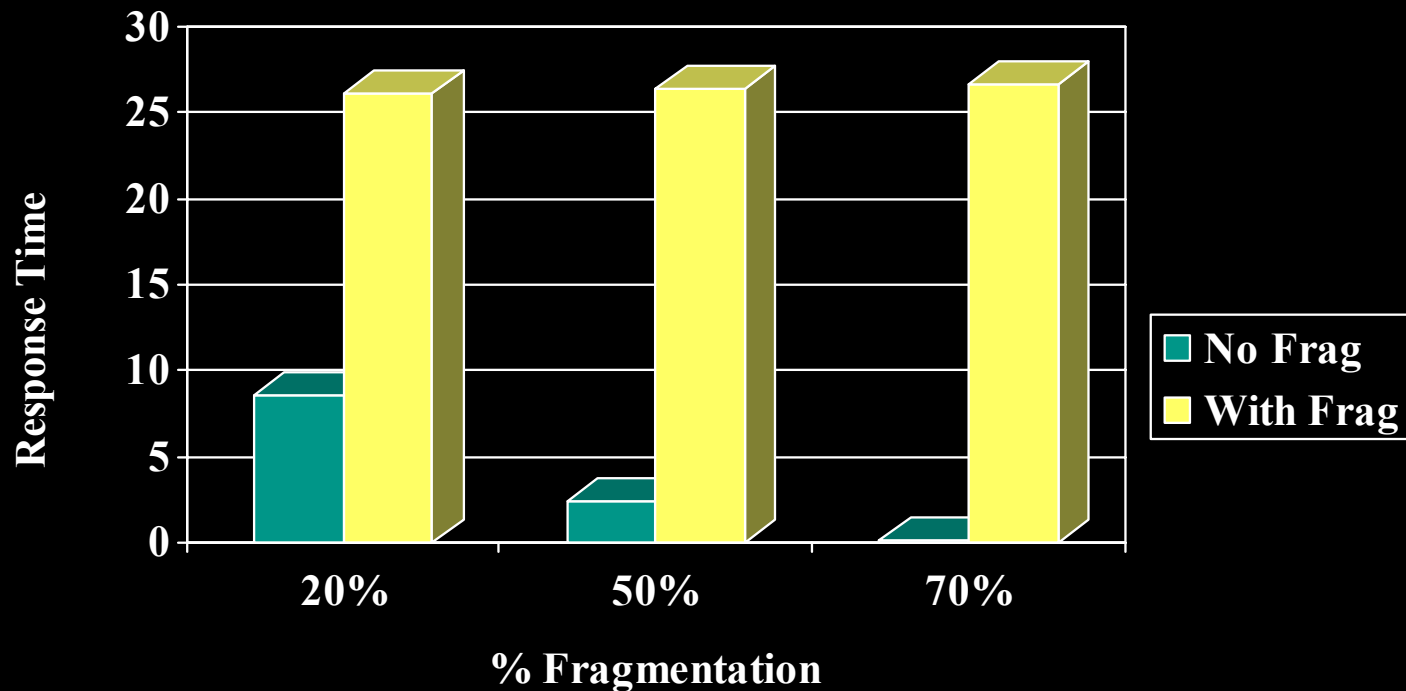
Internal Fragmentation

- Causes
 - Heap segment:
 - Bad choice of PCTFREE, PCTUSED
 - LOB segment
 - Bad choice of PCTVERSION, RETENTION
 - Most data inserted into segment is deleted
 - Steady state density of data in segment is small
 - Queue like behavior with temporal data in heap segments
 - Application typically does direct loads followed by deletes
 - Index segments with random updates & deletes with no further inserts
- Impact
 - Poor segment space utilization in the segment
 - Performance of full table scan, range scan, fast full scan etc., deteriorates by close to 100%

Internal Fragmentation

- Performance Impact
 - Internal fragmentations slows certain access paths, e.g., full table scan, fast full scan, etc.,

Full Table Scan Performance Degradation



Internal Fragmentation: Solution

- Online Segment Shrink remedies internal fragmentation
 - ROW MOVEMENT must be **ENABLED** for heap organized segments
 - Segment must be in ASSM, locally managed tablespace

Property	Shrink	Online Redef	Alter MOVE
Online	Y	Y	N
In-place	Y	N	N
Incremental	Y	N	N
Dependency Maintenance	Y	N	N
Segment Level Reorg	Y	N	Y
Parallel	N	Y	Y

- Note: Tables with large number of indexes, reorg is faster

Internal Fragmentation: Solution

- When to use Online Segment Shrink?
 - Not all fragmented segments are cause for concern
 - **Segment Advisor** is fragmentation advisor
 - Recommends when/how to defragment segment
 - Considers segment growth trend
 - Recommendations available through EM and advisor framework tables

Automatic Segment Advisor

- Introduced in Oracle Database 10g Release 2
- Enabled out-of-the-box
- Runs in default maintenance window
- Evaluates segments for fragmentation and makes appropriate recommendations
- Gives priority to segments that are:
 - Heavily accessed
 - In tablespaces with space alerts

Automate Segment Advisor in R1

- Create Segment Advisor job to run in maintenance window

```
variable id number;
begin
declare
  name varchar2(100) ; descr varchar2(500) ; objid number;
begin
  name := " ;
  descr := 'Segment Advisor Demo';
  dbms_advisor.create_task('Segment Advisor', :id, name, descr, NULL) ;
  dbms_advisor.create_object(name, 'TABLESPACE','USERS',NULL, NULL, NULL, objid);
  dbms_advisor.set_task_parameter(name, 'RECOMMEND_ALL', 'TRUE') ;
  dbms_advisor.execute_task(name) ;
end ;
end ;
/

set echo off
spool shrink.sql
select attr1 || ';' from dba_advisor_actions where task_id=:id ;
spool off
@shrink
```

Best Practice

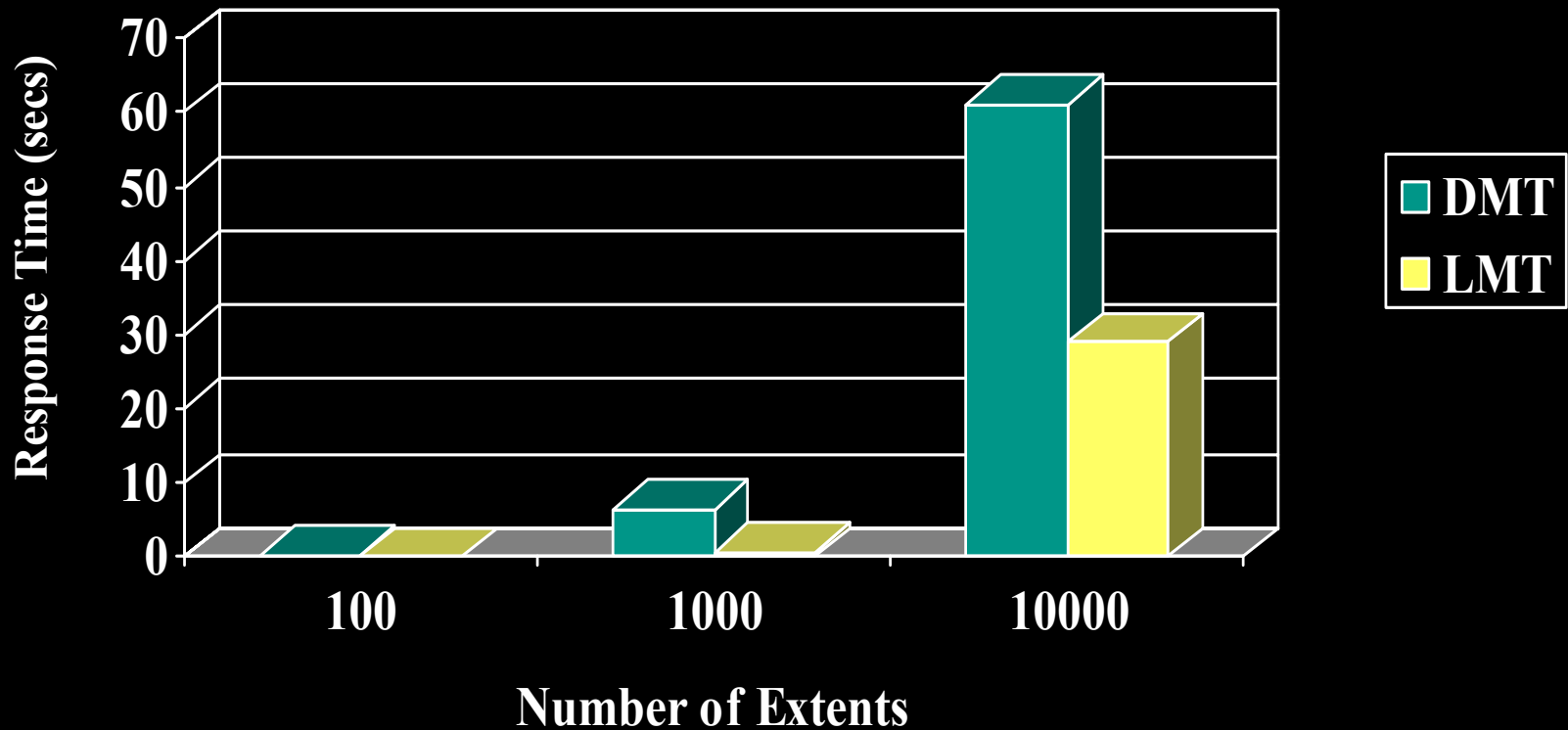
- Use Online Segment Shrink to eliminate internal fragmentation
- Evaluate Automatic Segment Advisor output periodically and implement recommendations as appropriate

Number of Extents

- Locally Managed Tablespaces
 - Extent map stored in metadata blocks
 - Extent map read during FTS, FFS, range scans
 - Non-issue because metadata addressability is very high
 - Four 8k blocks for 100G segment
 - Time spent reading metadata negligible
 - Number of extents an issue when
 - Performance of parallel queries is critical and number of extents exceed several hundred thousands and extents not coalescable
- Dictionary Managed Tablespaces (DMT)
 - Number of extents matter when performance of DDLs in DMT is critical; number of extents greater than 4096
- Best Practice
 - Use locally managed tablespace (**Recommended**)
 - See LMT vs. DMT Drop Performance comparison (next slide)
 - Alternatively: Keep extents in DMT below 1024-4096

LMT vs. DMT Performance Comparison

Drop Performance



Size of Extents

- Extents allow prefetch of data blocks
 - Very small extents will negatively impact I/O performance
 - Very large extents can result in space wastage
- Best Practice
 - Use auto-allocate with LMT
 - If using uniform, use following guideline

Tablespace Segments Size	Extent Size
< 1 M	64 KB
< 64 M	1 MB
< 1GB	16 MB
> 1 GB	100 MB

- Avoid large extents when segment merge loads common
 - Non-issue for auto-allocate

Default Permanent Tablespace

- SYSTEM is **database** default tablespace for storing permanent objects
- If **user default** ts not specified, SYSTEM can fill up and impact availability of entire database
- Unnecessary objects in SYSTEM can impact performance of operations that happen at file or tablespace level
- To avoid this ...



Best Practice: Designate Default Permanent Tablespace

Temporary Tablespace Management Best Practices

Temporary Tablespace Management

- Temporary data generated by database includes bitmap merges, hash join, bitmap index creation, sort, global temp tables etc.
- Data persists for duration of transaction or session
- High concurrency of space management operations is very critical
- Media and instance recovery is not required for objects in temporary tablespace

Best Practice

- Use locally managed **temporary** tablespace
 - Allows high concurrency space management
 - In steady state all space metadata cached in SGA
 - Operations serialized by SGA latch instead of database wide ST enqueue
 - Redo generated on temp blocks not written to disk allowing for faster writes
 - No files outside temp files modified — making possible Read Only database

Best Practice

- Locally managed temporary tablespaces are uniform extent tablespaces
- Guidelines for choosing extent size
 - 1M-10M:
 - For DSS, OLAP applications involving huge sorts, hash joins
 - Large temporary lobs are predominant
 - 64K or multiple:
 - Global temporary tables are predominant and amount of data loaded is small
 - Application is predominantly OLTP
- V\$tempseg_usage can be used to monitor space usage and workload distribution

SESSION_NUM	USERNAME	SEGTYPE	BLOCKS	TABLESPACE
101	SCOTT	SORT	128	TEMP
102	SCOTT	LOB_DATA	128	TEMP
103	SYS	HASH	256	TEMP

RAC Temporary Space Management

- Use single temporary tablespace for entire database
- No special configuration is needed
- Instances have dynamic affinity for space
- Each instance caches extents it has affinity to in its SGA
- Sharing of space between instances happens transparently and dynamically
 - Sharing is an expensive operation.
 - Add space when number of waits on SS enqueue increases

Temporary Tablespace Group

- One or more temporary tablespace can be assigned to a tablespace group
- Temporary tablespace group increases addressability from TB to PB
 - Note: Single temporary segment in the group cannot span multiple tablespaces
- User or a database can be assigned to an entire group

Undo Management Best Practices

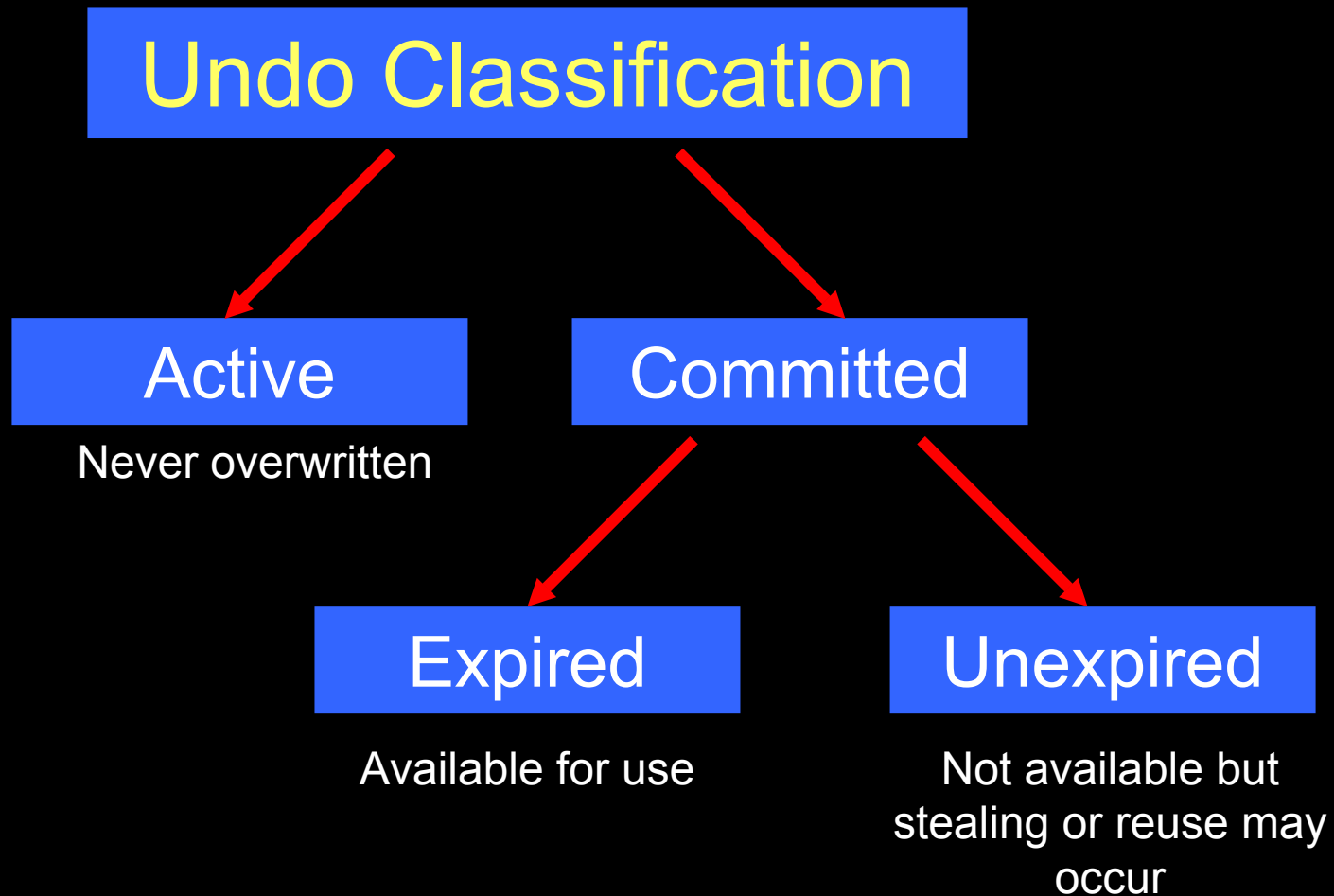
Manual vs. AUM

Description	Manual	AUM
Administration:		
Create tablespace	Yes	Yes
Specify initialization parameters	Yes	Yes
Create undo (rollback) segments	Yes	No
Determine segment number	Yes	No
Manage space/extent	Yes	No
Benefits:		
Dynamic: Adapts to changing workload	No	Yes
Automatic performance tuning	No	Yes
Predictable consistent-read	No	Yes
Predictable Flashback feature usage	No	Yes
Special application coding	Yes	No

AUM Administration

Steps	Description	9.1, 9.2	10.1	10.2
1	Create Undo tablespace	Yes	Yes	Yes
2	Set UNDO_MANAGEMENT=AUTO	Yes	Yes	Yes
3	Set UNDO_RETENTION	Yes	Optional	No

Undo Retention: Overview



Undo Retention: Overview

- Unexpired undo reuse/stealing – tablespace too small
 - Stealing: undo extents of other segments used
 - Reuse: undo extents of same segment used
 - `Select UNEXPSTEALCNT, UNXPBLKREUCNT from V$UNDOSTAT; (WRH$_UNDOSTAT)`
 - `V$UNDOSTAT`: 10 minute snapshots for 4 days
 - `WRH$_UNDOSTAT`: based on AWR retention policy (7 days by default)
- Undo “Retention Guarantee”
 - Unexpired undo never overwritten – transactions may fail
 - Tablespace attribute set using `ALTER TABLESPACE` command
 - Guaranteed Flashback usage

Configuring Undo Retention: Oracle 9i

- UNDO_RETENTION specifies time for which undo is considered unexpired
- Minimum value should be set to length of longest running query
 - `Select max(maxquerylen) from v$undostat;`
- Used for predictable Flashback Query usage
- Current undo retention value can be obtained from TUNED_UNDORETENTION column of V\$UNDOSTAT



Best Practice: Set to higher of longest query length or
Flashback Query needs

Configuring Undo Retention: Oracle 10g Release 1 (10.1)

- Undo retention is auto-tuning
 - Oracle tracks active queries to keep undo retention ahead of longest running query
 - Undo generation rate growth projected forward
 - Adjusts to changing workloads, e.g., OLTP to DSS
 - More efficient space utilization: no space wastage by keeping unnecessary undo
- UNDO_RETENTION
 - specifies lower limit (floor) for undo retention
 - Used for Flashback features



Best Practice: Set to Flashback needs

Configuring Undo Retention: Oracle 10g Release 2 (10.2)

- Undo retention auto-tuning further enhanced
 - Tuned for maximum retention possible for given tablespace
 - Maximum retention available for fixed ts (85% of size)
 - For auto-extensible tablespace, same behavior as in 10.1
 - Adjusts automatically different workloads
 - Efficient space utilization
- UNDO_RETENTION
 - Specifies lower limit retention for auto-extensible ts
 - Used for Flashback purposes when GUARANTEE enabled



Best Practice: Don't set UNDO_RETENTION

Undo Tablespace: Overview

- Special tablespace for storing undo information
- Created as part of CREATE DATABASE command

```
CREATE DATABASE orcldb
LOGFILE GROUP 1 ('redo01.log') SIZE 500M
...
...
UNDO TABLESPACE undo_tbs
DATAFILE 'undotbs01.dbf'
SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE
UNLIMITED;
```

Undo Tablespace: Overview

- Or as separate CREATE TABLESPACE command

```
CREATE UNDO TABLESPACE undo_tbs  
DATAFILE 'undo.dbf' SIZE 500M REUSE AUTOEXTEND ON;
```

- One active Undo tablespace per DB
- Except for RAC
 - One Undo tablespace per instance
 - Undo generated from instance stored in own tablespace
 - Any instance can read data from any Undo tablespace for consistent-read purposes
 - An instance can also update another Undo tablespace for transaction recovery purposes

Configuring Undo Tablespace

- Initial Setting: New System
 - Undo generation statistics not available
 - Undo Advisor not very accurate
 - Enable AUTOEXTEND datafile attribute
 - Auto-tuning of undo retention will grow tablespace as needed
- Final Setting: Steady State
 - Disable AUTOEXTEND to isolate from runaway queries
 - Run Undo Advisor to determine tablespace size
 - Add 20% (safety margin) and fix tablespace size

Undo Advisor

Advisor

* New Undo Retention minutes

Analysis Time Period

Choose the time period that best represents system activity

[Update Analysis and Graph](#)

Selected Analysis Time Period **11/8/04 4:00 PM - 11/15/04 4:00 PM**

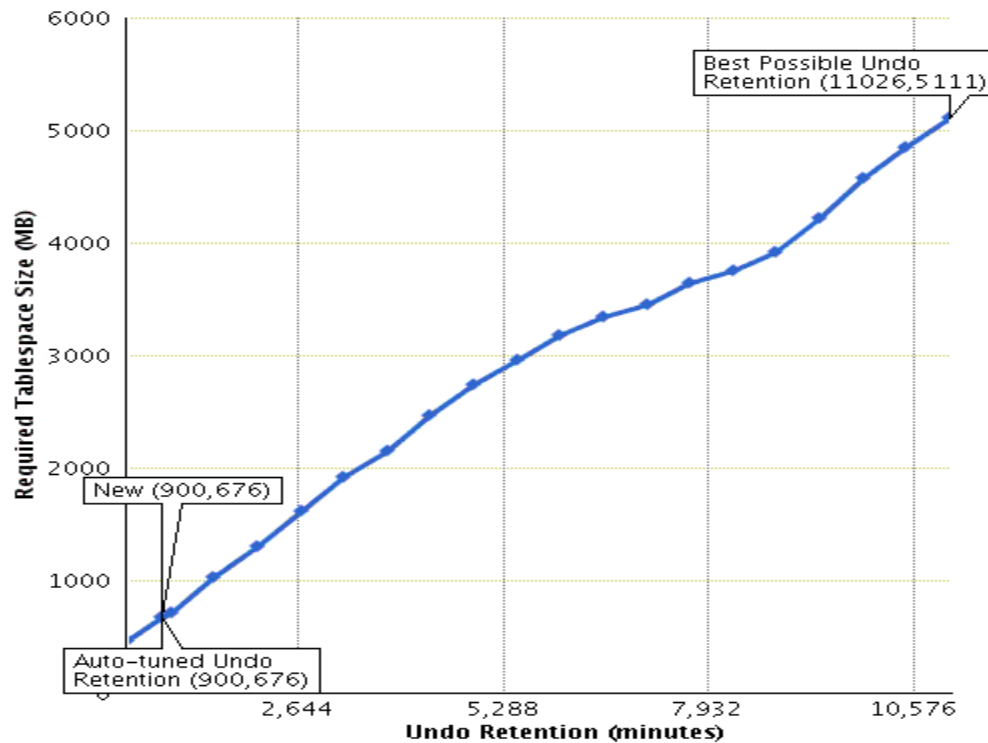
Analysis

Required Tablespace Size for New Undo Retention (MB) **676**

Minimum Undo Retention (minutes) **900**

Best Possible Undo Retention (minutes) **11026**

Required Tablespace Size by Undo Retention Length



Problem Prevention and Automatic Exception Handling

Problem Prevention and Automatic Exception Handling

- Server Generated Alerts use Advanced Queues (AQ) infrastructure
- Use AQ and alerts to further automate space & undo management
- This is a 3 step process
 - Subscribe to ALERT_QUEUE
 - Create a callback procedure that handles the exception in question. Procedure maybe PL/SQL, JMS, or OCI function
 - Register the callback procedure

Step 1: Alert Queue Subscription

Space Alerts

```
DECLARE
subscriber      aq$_agent;
BEGIN
subscriber := aq$_agent('SPACE_SUB', 'ALERT_QUE', null);
dbms_aqadm.add_subscriber(
    queue_name => 'ALERT_QUE',
    subscriber => subscriber,
    rule       => 'tab.user_data.reason_id = 9 and
tab.user_data.object_name = ''TEST_TBS'' ');
END;
/
```

Alerts when tablespace is full

Step 2: Callback Procedure Definition

```
create or replace procedure myCALLBACK (context raw, reginfo sys.aq$_reg_info,  
                                         descr sys.aq$_descriptor, payload raw, payloadl number)  
AS  
  dequeue_options  DBMS_AQ.dequeue_options_t;  
  message_properties DBMS_AQ.message_properties_t;  
  message_handle RAW(16); message ALERT TYPE;  
BEGIN  
  dequeue_options.consumer_name := 'SPACE_SUB' ;  
  -- Dequeue the message  
  DBMS_AQ.DEQUEUE(queue_name => 'SYS.ALERT_QUE',  
                  dequeue_options => dequeue_options,  
                  message_properties => message_properties,  
                  payload => message, msgid => message_handle);  
  commit,  
  -- provide your remedy PL/SQL code here, e.g., run Segment Shrink.  
END;  
/
```

Step 3: Callback Procedure Registration

```
DECLARE
reginfo1 sys.aq$_reg_info;
reginfo1_list sys.aq$_reg_info_list;

BEGIN
reginfo1 := sys.aq$_reg_info('SYS.ALERT_QUEUE:SPACE_SUB',
DBMS_AQ.NAMESPACE_AQ, 'plsql://SYS.MYCALLBACK',
HEXTORAW('FF'));
-- Create the registration info list
reginfo1_list := sys.aq$_reg_info_list(reginfo1);
sys.dbms_aq.register(reginfo1_list, 1);
END;
/
```

Space Problem Prevention/ Handling

- Prevent/Handle space exceptions using
 - Oracle Managed Files (OMF) to facilitate space management
 - Run Segment Advisor
 - Perform Online Segment Shrink
 - ALTER TABLE <table_name> SHRINK SPACE**
 - Add datafile

Segment Advisor/Shrink Automation

```
variable id number;
begin
declare
  name varchar2(100) ; descr varchar2(500) ; objid number;
begin
  name := " ;
  descr := 'Segment Advisor Demo';
  dbms_advisor.create_task('Segment Advisor', :id, name, descr, NULL) ;
  dbms_advisor.create_object(name, 'TABLESPACE','USERS',NULL, NULL, NULL, objid);
  dbms_advisor.set_task_parameter(name, 'RECOMMEND_ALL', 'TRUE') ;
  dbms_advisor.execute_task(name) ;
end ;
end ;
/

set echo off
spool shrink.sql
select attr1 || ';' from dba_advisor_actions where task_id=:id ;
spool off
@shrink
```

Undo Problem Prevention/ Handling

- Subscribe to Long Query Alert (ORA-1555)
- When alert occurs:
 - Run Undo Advisor
 - Implement recommendation automatically

```
declare
  retention number ; undo_ts_size number ;
begin
  retention := DBMS_UNDO_ADV.REQUIRED_RETENTION ;
  undo_ts_size := DBMS_UNDO_ADV.REQUIRED_UNDO_SIZE(retention) ;
  dbms_output.put_line('Required tablespace size is ' || undo_ts_size); end ;
/
```

Note: More sophisticated sample code in white paper

Conclusion

Conclusion



- Effective space and undo management
 - Eliminates/minimizes fragmentation
 - Enhances performance and availability
 - Optimizes space utilization
 - Eliminates ORA-1555
 - Enables predictable Flashback feature usage

Simplifies the DBA's job!

A large, stylized graphic of the letters 'Q' and 'A' in a dark grey, serif font. A large, bright red ampersand is superimposed over the 'Q' and 'A'.

**QUESTIONS
ANSWERS**

ORACLE
OPEN
WORLD

ORACLE®